# pywws Documentation

***Version 13.09***$_r$1068

**Jim Easterbrook**

10 September 2015

Table des matières

Logiciel Python pour Station météo USB sans-fil
http://pythonhosted.org/pywws
http://jim-easterbrook.github.com/pywws/
http://github.com/jim-easterbrook/pywws/

Ce document est disponible dans les langues suivantes (Les versions autres que anglais peuvent ne pas être complètes ou à jour) :

- English
- Français
- Italiano

CHAPITRE 1

# Introduction

pywws est une collection de scripts Python pour lire, stocker et traiter les données des stations météorologiques sans fil USB courantes tels que Elecsa AstroTouch 6975, Watson W-8681, WH-1080PC, WH1080, WH1081, WH3080 etc. Je suppose que tout modèle de station équipée du logiciel EasyWeather pour Windows est compatible, sans pouvoir le garantir.

Le logiciel a été développé pour fonctionner dans environnement de faible puissance, avec peu de mémoire tel un routeur. Il peut être utilisé pour créer des graphiques et des pages Web affichant de récentes lectures météorologiques, généralement mis à jour à chaque heure. Il peut également envoyer des données à des services tels que : Weather Underground http://www.wunderground.com/ et poster des messages sur Twitter.

J'ai écrit ce logiciel pour répondre à mes besoins, mais ai essayé de le rendre adaptable aux besoins des autres. Vous voudrez peut-être modifier certains ou tous les modules, ou en écrire de nouveaux, pour lui faire faire exactement ce que vous souhaitez. L'une des raisons pour lesquelles Python est utilisé est qu'il rend de telles modifications si facile. N'ayez pas peur, essayez-le vous verrez..

CHAPITRE 2

# Exigences

The software you'll need to run pywws depends on what you plan to do with it. You'll need Python 2.5 or later – Python 3 is partially supported, some functionality depends on libraries that have not yet been ported to Python 3.

For more detail, see Pré-requis.

CHAPITRE 3

# Obtenir une copie de pywws

La façon la plus simple d'obtenir pywws est de télécharger le fichier zip ou tar.gz depuis l'Index des paquets Python (PyPI) et ensuite extraire les fichiers dans un répertoire distinct sur votre ordinateur. Ces fichiers 'archives' contiennent la toute dernière version du logiciel - une nouvelle version sort tous les quelques mois.

```
sudo pip install pywws
```

Si vous n'avez pas les privilèges sur la racine, ou ne désirez pas installer pywws, vous pouvez télécharger le fichier zip ou .tar.gz à partir de PyPI et extraire les fichiers dans un dossier de votre choix, sur votre ordinateur

Les fichiers PyPI contiennent un instantané de la version du logiciel - une nouvelle version est publiée tous les quelques mois. Si vous souhaitez être au fait des derniers développements de pywws, vous pouvez utiliser `git` pour cloner le répertoire pywws :

```
git clone https://github.com/jim-easterbrook/pywws.git
```

Par la suite, vous pouvez compiler la documentation et les fichiers de localisation de langue (ce qui nécessite les logiciels `gettext` et `sphinx`) :

```
cd pywws
python setup.py build_sphinx
python setup.py msgfmt
```

Pour plus de détails, voir Comment démarrer avec pywws.

## 3.1 Mettre à jour pywws

La méthode employée pour faire la mise à jour de pywws dépend de la façon dont vous l'avez à l'origine obtenu. Si vous avez téléchargé un fichier zip ou tar.gz, vous devez procédé de la même manière avec la nouvelle version, puis supprimez le téléchargement obsolète quand vous avez terminé l'installation de la nouvelle version. (Notez que l'évolution est beaucoup plus facile si vous n'entreposez pas vos fichiers de calibration, modules utilisateur et données météo dans le même répertoire que les fichiers téléchargés.) Si vous aviez utilisé `pip` vous n'avez qu'à répéter la commande. Les utilisateurs de `git` n'ont qu'à faire la commande `git pull`.

Certaines récentes versions de pywws ont changé ce qui est stocké dans les fichiers de données horaires, quotidiens ou mensuels. Ces nouvelles versions sont incompatibles avec les données traitées par les versions antérieures. : le module *`pywws.Reprocess`* régénère toutes les données récapitulatives. Il devrait être exécuté après chaque mise à jour principale

CHAPITRE 4

# Documentation

La documentation est inclue avec le téléchargement de pywws, et est également disponible en ligne. Un bon point de départ est le Guide de démarrage qui décrit plus en détail comment installer pywws.

Si vous avez des questions dont vous ne trouvez pas réponse dans la documentation, s'il vous plaît joindre la liste / groupe de discussion pywws Google et y poser votre question. Notez que votre premier message n'apparaît pas immédiatement – les nouveaux membres doivent être approuvés par un modérateur, pour empêcher les pourriels.

## 4.1 Contenu

### 4.1.1 Licence Publique Générale GNU

```
                    GNU GENERAL PUBLIC LICENSE
                       Version 2, June 1991

 Copyright (C) 1989, 1991 Free Software Foundation, Inc.
     51 Franklin Street, Fifth Floor, Boston, MA  02110-1301  USA
 Everyone is permitted to copy and distribute verbatim copies
 of this license document, but changing it is not allowed.

                            Preamble

  The licenses for most software are designed to take away your
freedom to share and change it.  By contrast, the GNU General Public
License is intended to guarantee your freedom to share and change free
software--to make sure the software is free for all its users.  This
General Public License applies to most of the Free Software
Foundation's software and to any other program whose authors commit to
using it.  (Some other Free Software Foundation software is covered by
the GNU Library General Public License instead.)  You can apply it to
your programs, too.

  When we speak of free software, we are referring to freedom, not
price.  Our General Public Licenses are designed to make sure that you
have the freedom to distribute copies of free software (and charge for
this service if you wish), that you receive source code or can get it
if you want it, that you can change the software or use pieces of it
in new free programs; and that you know you can do these things.

  To protect your rights, we need to make restrictions that forbid
anyone to deny you these rights or to ask you to surrender the rights.
```

```
These restrictions translate to certain responsibilities for you if you
distribute copies of the software, or if you modify it.

  For example, if you distribute copies of such a program, whether
gratis or for a fee, you must give the recipients all the rights that
you have.  You must make sure that they, too, receive or can get the
source code.  And you must show them these terms so they know their
rights.

  We protect your rights with two steps: (1) copyright the software, and
(2) offer you this license which gives you legal permission to copy,
distribute and/or modify the software.

  Also, for each author's protection and ours, we want to make certain
that everyone understands that there is no warranty for this free
software.  If the software is modified by someone else and passed on, we
want its recipients to know that what they have is not the original, so
that any problems introduced by others will not reflect on the original
authors' reputations.

  Finally, any free program is threatened constantly by software
patents.  We wish to avoid the danger that redistributors of a free
program will individually obtain patent licenses, in effect making the
program proprietary.  To prevent this, we have made it clear that any
patent must be licensed for everyone's free use or not licensed at all.

  The precise terms and conditions for copying, distribution and


    GNU GENERAL PUBLIC LICENSE
   TERMS AND CONDITIONS FOR COPYING, DISTRIBUTION AND MODIFICATION
  0. This License applies to any program or other work which contains
a notice placed by the copyright holder saying it may be distributed
under the terms of this General Public License.  The "Program", below,
refers to any such program or work, and a "work based on the Program"
means either the Program or any derivative work under copyright law:
that is to say, a work containing the Program or a portion of it,
either verbatim or with modifications and/or translated into another
language.  (Hereinafter, translation is included without limitation in
the term "modification".)  Each licensee is addressed as "you".
Activities other than copying, distribution and modification are not
covered by this License; they are outside its scope.  The act of
running the Program is not restricted, and the output from the Program
is covered only if its contents constitute a work based on the
Program (independent of having been made by running the Program).
Whether that is true depends on what the Program does.
  1. You may copy and distribute verbatim copies of the Program's
source code as you receive it, in any medium, provided that you
conspicuously and appropriately publish on each copy an appropriate
copyright notice and disclaimer of warranty; keep intact all the
notices that refer to this License and to the absence of any warranty;
and give any other recipients of the Program a copy of this License
along with the Program.
You may charge a fee for the physical act of transferring a copy, and
```

```
you may at your option offer warranty protection in exchange for a fee.
  2. You may modify your copy or copies of the Program or any portion
of it, thus forming a work based on the Program, and copy and
distribute such modifications or work under the terms of Section 1
above, provided that you also meet all of these conditions:
    a) You must cause the modified files to carry prominent notices
    stating that you changed the files and the date of any change.
    b) You must cause any work that you distribute or publish, that in
    whole or in part contains or is derived from the Program or any
    part thereof, to be licensed as a whole at no charge to all third
    parties under the terms of this License.
    c) If the modified program normally reads commands interactively
    when run, you must cause it, when started running for such
    interactive use in the most ordinary way, to print or display an
    announcement including an appropriate copyright notice and a
    notice that there is no warranty (or else, saying that you provide
    a warranty) and that users may redistribute the program under
    these conditions, and telling the user how to view a copy of this
    License.  (Exception: if the Program itself is interactive but
    does not normally print such an announcement, your work based on
    the Program is not required to print an announcement.)
These requirements apply to the modified work as a whole.  If
identifiable sections of that work are not derived from the Program,
and can be reasonably considered independent and separate works in
themselves, then this License, and its terms, do not apply to those
sections when you distribute them as separate works.  But when you
distribute the same sections as part of a whole which is a work based
on the Program, the distribution of the whole must be on the terms of
this License, whose permissions for other licensees extend to the
entire whole, and thus to each and every part regardless of who wrote it.
Thus, it is not the intent of this section to claim rights or contest
your rights to work written entirely by you; rather, the intent is to
exercise the right to control the distribution of derivative or
collective works based on the Program.
In addition, mere aggregation of another work not based on the Program
with the Program (or with a work based on the Program) on a volume of
a storage or distribution medium does not bring the other work under
the scope of this License.
  3. You may copy and distribute the Program (or a work based on it,
under Section 2) in object code or executable form under the terms of
Sections 1 and 2 above provided that you also do one of the following:
    a) Accompany it with the complete corresponding machine-readable
    source code, which must be distributed under the terms of Sections
    1 and 2 above on a medium customarily used for software interchange; or,
    b) Accompany it with a written offer, valid for at least three
    years, to give any third party, for a charge no more than your
    cost of physically performing source distribution, a complete
    machine-readable copy of the corresponding source code, to be
    distributed under the terms of Sections 1 and 2 above on a medium
    customarily used for software interchange; or,
    c) Accompany it with the information you received as to the offer
```

```
    to distribute corresponding source code.  (This alternative is
    allowed only for noncommercial distribution and only if you
    received the program in object code or executable form with such
    an offer, in accord with Subsection b above.)
The source code for a work means the preferred form of the work for
making modifications to it.  For an executable work, complete source
code means all the source code for all modules it contains, plus any
associated interface definition files, plus the scripts used to
control compilation and installation of the executable.  However, as a
special exception, the source code distributed need not include
anything that is normally distributed (in either source or binary
form) with the major components (compiler, kernel, and so on) of the
operating system on which the executable runs, unless that component
itself accompanies the executable.
If distribution of executable or object code is made by offering
access to copy from a designated place, then offering equivalent
access to copy the source code from the same place counts as
distribution of the source code, even though third parties are not
compelled to copy the source along with the object code.
  4. You may not copy, modify, sublicense, or distribute the Program
except as expressly provided under this License.  Any attempt
otherwise to copy, modify, sublicense or distribute the Program is
void, and will automatically terminate your rights under this License.
However, parties who have received copies, or rights, from you under
this License will not have their licenses terminated so long as such
parties remain in full compliance.
  5. You are not required to accept this License, since you have not
signed it.  However, nothing else grants you permission to modify or
distribute the Program or its derivative works.  These actions are
prohibited by law if you do not accept this License.  Therefore, by
modifying or distributing the Program (or any work based on the
Program), you indicate your acceptance of this License to do so, and
all its terms and conditions for copying, distributing or modifying
the Program or works based on it.
  6. Each time you redistribute the Program (or any work based on the
Program), the recipient automatically receives a license from the
original licensor to copy, distribute or modify the Program subject to
these terms and conditions.  You may not impose any further
restrictions on the recipients' exercise of the rights granted herein.
You are not responsible for enforcing compliance by third parties to
this License.
  7. If, as a consequence of a court judgment or allegation of patent
infringement or for any other reason (not limited to patent issues),
conditions are imposed on you (whether by court order, agreement or
otherwise) that contradict the conditions of this License, they do not
excuse you from the conditions of this License.  If you cannot
distribute so as to satisfy simultaneously your obligations under this
License and any other pertinent obligations, then as a consequence you
may not distribute the Program at all.  For example, if a patent
license would not permit royalty-free redistribution of the Program by
all those who receive copies directly or indirectly through you, then
```

```
the only way you could satisfy both it and this License would be to
refrain entirely from distribution of the Program.
If any portion of this section is held invalid or unenforceable under
any particular circumstance, the balance of the section is intended to
apply and the section as a whole is intended to apply in other
circumstances.
It is not the purpose of this section to induce you to infringe any
patents or other property right claims or to contest validity of any
such claims; this section has the sole purpose of protecting the
integrity of the free software distribution system, which is
implemented by public license practices.  Many people have made
generous contributions to the wide range of software distributed
through that system in reliance on consistent application of that
system; it is up to the author/donor to decide if he or she is willing
to distribute software through any other system and a licensee cannot
impose that choice.
This section is intended to make thoroughly clear what is believed to
be a consequence of the rest of this License.
  8. If the distribution and/or use of the Program is restricted in
certain countries either by patents or by copyrighted interfaces, the
original copyright holder who places the Program under this License
may add an explicit geographical distribution limitation excluding
those countries, so that distribution is permitted only in or among
countries not thus excluded.  In such case, this License incorporates
the limitation as if written in the body of this License.
  9. The Free Software Foundation may publish revised and/or new versions
of the General Public License from time to time.  Such new versions will
be similar in spirit to the present version, but may differ in detail to
address new problems or concerns.
Each version is given a distinguishing version number.  If the Program
specifies a version number of this License which applies to it and "any
later version", you have the option of following the terms and conditions
either of that version or of any later version published by the Free
Software Foundation.  If the Program does not specify a version number of
this License, you may choose any version ever published by the Free Software
Foundation.
  10. If you wish to incorporate parts of the Program into other free
programs whose distribution conditions are different, write to the author
to ask for permission.  For software which is copyrighted by the Free
Software Foundation, write to the Free Software Foundation; we sometimes
make exceptions for this.  Our decision will be guided by the two goals
of preserving the free status of all derivatives of our free software and
of promoting the sharing and reuse of software generally.



    NO WARRANTY
  11. BECAUSE THE PROGRAM IS LICENSED FREE OF CHARGE, THERE IS NO WARRANTY
FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW.  EXCEPT WHEN
OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES
PROVIDE THE PROGRAM "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED
```

```
OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF
MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE.  THE ENTIRE RISK AS
TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU.  SHOULD THE
PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING,
REPAIR OR CORRECTION.
  12. IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING
WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MAY MODIFY AND/OR
REDISTRIBUTE THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES,
INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING
OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED
TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY
YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER
PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE
POSSIBILITY OF SUCH DAMAGES.


     END OF TERMS AND CONDITIONS

    How to Apply These Terms to Your New Programs
  If you develop a new program, and you want it to be of the greatest
possible use to the public, the best way to achieve this is to make it
free software which everyone can redistribute and change under these terms.
  To do so, attach the following notices to the program.  It is safest
to attach them to the start of each source file to most effectively
convey the exclusion of warranty; and each file should have at least
the "copyright" line and a pointer to where the full notice is found.
    <one line to give the program's name and a brief idea of what it does.>
    Copyright (C) <year>  <name of author>
    This program is free software; you can redistribute it and/or modify
    it under the terms of the GNU General Public License as published by
    the Free Software Foundation; either version 2 of the License, or
    (at your option) any later version.
    This program is distributed in the hope that it will be useful,
    but WITHOUT ANY WARRANTY; without even the implied warranty of
    MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
    GNU General Public License for more details.
    You should have received a copy of the GNU General Public License
    along with this program; if not, write to the Free Software
    Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA  02110-1301  USA
Also add information on how to contact you by electronic and paper mail.
If the program is interactive, make it output a short notice like this
when it starts in an interactive mode:
    Gnomovision version 69, Copyright (C) year  name of author
    Gnomovision comes with ABSOLUTELY NO WARRANTY; for details type `show w'.
    This is free software, and you are welcome to redistribute it
    under certain conditions; type `show c' for details.
The hypothetical commands `show w' and `show c' should show the appropriate
parts of the General Public License.  Of course, the commands you use may
be called something other than `show w' and `show c'; they could even be
mouse-clicks or menu items--whatever suits your program.
You should also get your employer (if you work as a programmer) or your
```

```
school, if any, to sign a "copyright disclaimer" for the program, if
necessary.  Here is a sample; alter the names:
  Yoyodyne, Inc., hereby disclaims all copyright interest in the program
  `Gnomovision' (which makes passes at compilers) written by James Hacker.
  <signature of Ty Coon>, 1 April 1989
  Ty Coon, President of Vice
This General Public License does not permit incorporating your program into
proprietary programs.  If your program is a subroutine library, you may
consider it more useful to permit linking proprietary applications with the
library.  If this is what you want to do, use the GNU Library General
Public License instead of this License.
modification follow.
```

### 4.1.2 Pré-requis

La liste des logiciel requis par pywws peut être effrayante au premier abord. Cependant, plusieurs de ces paquets ne sont pas nécessaires pour la plupart des utilisateurs. Ce que vous avez besoin dépend de ce que vous voulez faire avec pywws. Rappelez-vous que pywws est "un ensemble de pièces" plutôt qu'une application monolithique.

Vous devriez être en mesure d'installer la plupart de ceux-ci en utilisant le gestionnaire de paquets de votre système d'exploitation ; ce qui est beaucoup plus facile que de télécharger et compiler à partir des fichiers source du site Web du projet. Notez que quelques distributions de Linux emploient différents noms pour certains paquets, par exemple pour Ubuntu, pyusb se nomme python-usb.

Alternativement, vous devriez pouvoir installer des versions plus récentes de certaines bibliothèques à partir du Python Package Index (PyPI). Je recommande d'installer **'pip < http ://www.pip-installer.org/ >'_** (le paquet peut s'appeler python-pip) ou le **'easy_install de < http ://peak.telecommunity.com/DevCenter/EasyInstall >'_**. Chacun d'eux simplifient l'installation de logiciel à partir de PyPI. Par exemple, pour installer PyUSB à partir de PyPI en utilisant la commande `pip` :

```
sudo pip install pyusb
```

#### Essentiel

— Python version 2.5 ou plus récent.

La version 3 de python est supportée, mais certains modules pourraient ne pas fonctionner correctement. Si vous trouvez un problème avec python 3, svp envoyez un message sur la **'liste pywws < http ://groups.google.com/group/pywws >'_** ou soumettez un rapport de bogue sur GitHub.

##### Librairie USB

Pour lire les données d'une station météorologique, pywws a besoin d'une bibliothèque python qui lui permet de communiquer par l'intermédiaire d'un port USB. Il y a un grand choix de bibliothèques USB qui peuvent être employées. Elles ne sont pas toutes disponibles sur toutes les plates-formes, ce qui peut limiter votre choix.

Sous MacOS X le pilote du système d'exploitation "réclame" la station météorologique, ce qui empêche le fonctionnement de libusb. Ceci limite les utilisateurs de Mac à l'option 3 ou 4.

— Librairies USB option 1 (préférable, sauf sous MacOS)
  — PyUSB version 1.0
  — libusb version 0.1 ou version 1.0
— Librairies USB option 2 (Si PyUSB 1.0 n'est pas disponible)

  - PyUSB version 0.4
  - libusb version 0.1
- Librairies USB option 3 (préférable pour MacOS)
  - hidapi
  - ctypes (included with many Python installations)
- Librairies USB option 4
  - hidapi
  - cython-hidapi
  - cython

### Graphes

Le module *pywws.Plot* utilise `gnuplot` pour générer les graphes. Si vous voulez produire des graphiques de données météo, par exemple pour inclure dans une page Web, vous devez installer l'application `gnuplot` :

- gnuplot v4.2 ou plus récent

### Téléversement web sécurisé (sftp)

Le module *pywws.Upload* peut utiliser "FTP sur ssh" (sftp) pour envoyer les fichiers sur votre site web. L'envoi normal emploi simplement les modules standard Python, mais si vous souhaitez utiliser sftp vous devez installer ces deux modules :

- paramiko
- pycrypto

### Mise à jour Twitter

The *pywws.ToTwitter* module can be used to send weather status messages to Twitter. Posting to Twitter requires all four of these modules :

- python-twitter v0.8.6 ou plus récent
- simplejson
- python-oauth2
- httplib2

Modifié dans la version 13.06_r1023 : pywws previously used the tweepy library instead of `python-twitter` and `python-oauth2`.

### Pour créer une nouvelle traduction

pywws peut être configuré pour utiliser d'autres langues que l'anglais, et la documentation peut aussi être traduite dans d'autres langues. Voir Comment utiliser pywws dans une autre langue pour plus d'informations. Le paquet `gettext` est nécessaire pour extraire les phrases pouvant être traduites, et pour compiler les fichiers de traduction.

- gettext

### Pour 'compiler' la documentation

La documentation de pywws est produite en "texte ReStructurées". Un programme appelé `Sphinx` est utilisé pour convertir ce format facile à écrire, au format HTML pour une utilisation avec un navigateur web. Si vous souhaitez créer une copie locale de la documentation (donc vous n'avez pas à dépendre de la version en ligne, ou pour tester une traduction que vous travaillez) vous devez installer `Sphinx`.

- sphinx

Commentaires et questions ? SVP, abonnez-vous à la liste pywws http://groups.google.com/group/pywws et faites-vous entendre.

### 4.1.3 Historique

```
pywws - Python software for USB Wireless Weather Stations
http://github.com/jim-easterbrook/pywws
Copyright (C) 2008-13  Jim Easterbrook  jim@jim-easterbrook.me.uk

This program is free software; you can redistribute it and/or
modify it under the terms of the GNU General Public License
as published by the Free Software Foundation; either version 2
of the License, or (at your option) any later version.

This program is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
GNU General Public License for more details.

You should have received a copy of the GNU General Public License
along with this program; if not, write to the Free Software
Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA  02110-1301, USA.

Changes in v13.06:
    1/ Substantially rearranged directories, getting rid of 'code' and 'code3'.
    2/ Removed 'makefile' - everything is now done via 'setup.py'.
    3/ Removed 'RunModule.py' - use 'python -m pywws.module' now.
    4/ Separated storage of config (weather.ini) and status (status.ini).
    5/ Replaced toservice.py "rapid fire" mode with a separate config file for
       Weather Underground rapid fire.
    6/ Added 2 more low-level USB access modules.
    7/ Various bug fixes and minor improvements.

Changes in v13.03:
    1/ Added 'rain days' to monthly data. (Reprocess required when upgrading.)
    2/ Extended template syntax to include comments.
    3/ Added 'humidity index' function.
    4/ Added French translation of documentation.
    5/ Reduced frequency of saving data files.
    6/ Various bug fixes.

Changes in v12.12:
    1/ Added support for Python 3.
    2/ Added French documentation translation.
    3/ Used 'binary search' to speed up data access.
    4/ Various bug fixes.

Changes in v12.11:
    1/ Moved development from Google code to GitHub.
    2/ Made software attempt to avoid USB activity at times when it is assumed
       the weather station might be writing to its memory. This might solve
       the USB lockup problem, but it's too early to tell.

Changes in v12.10:
    1/ Added a 'winddir_text' function for use in templates.
    2/ Added <ytics> and <y2tics> options to graph plots.
    3/ Various bug fixes.

Changes in v12.07:
    1/ Added Open Weather Map to the services.
    2/ Fixed problem with Weather Underground uploads that started on 1st June.
```

```
    3/ Various bug fixes and software structure improvements.

Changes in v12.05:
    1/ Made 'fixed block' data available to template calculations.
    2/ Fixed buggy auto-detection of 3080 weather stations.
    3/ Added a function to generate the Zambretti forecast code letter.
    4/ Added a program to test USB communication reliablility.
    5/ Various bug fixes and software structure improvements.

Changes in v12.02:
    1/ Separated out low level USB communications to enable use of different
       libraries. Now works on recent versions of Mac OS.
    2/ Added humidity, pressure & wind data to summary data.
    3/ Merged Weather Underground and UK Met Office uploaders into one combined
       module. Added more 'service' uploaders.
    4/ Various bug fixes and software structure improvements.

Changes in v11.10:
    1/ Complete restructuring of documentation.
    2/ Added a user defined 'calibration' process.
    3/ Sets 'locale' according to language setting.
    4/ Added ability to upload to UK Met Office 'WOW'.
    5/ Various bug fixes and software structure improvements.
    6/ New language files: French, Danish.

Changes in v11.05:
    1/ Added support for '3080' family stations that have illuminance and
       UV sensors.
    2/ Broadened the range of tasks that can be done with 'live' data.
    3/ Various bug fixes and software structure improvements.

Changes in v11.02:
    1/ Various bug fixes and software structure improvements.
    2/ Improved wind direction averaging.
    3/ Added conversion functions for common things such as C to F.
    4/ Added a YoWindow module.
    5/ Improved Zambretti forecaster.

Changes in v10.12:
    1/ Various bug fixes and software structure improvements.
    2/ Added a 'goto' instruction to Template.py.
    3/ Added a 'Zambretti' forecast function to Template.py. This should
       be treated as an experiment, and not relied upon for accuracy.

Changes in v10.10:
    1/ Added 'catchup' mode to ToUnderground.py.
    2/ Created 'Tasks.py' to handle common tasks.
    3/ Made better use of Python's logger for info and error
       messages.
    4/ Changed over from 'python-twitter' to 'tweepy' for Twitter
       access. Twitter authorisation using OAuth now works.
    5/ Added 'LiveLog.py' live logging program.
    6/ Added 'SetWeatherStation.py' to do some configuration of weather
       station. No longer need EasyWeather to set logging interval!
    7/ Added 'Rapid Fire' ability to ToUnderground.py.
    8/ Added plain text versions of HTML documentation.
    9/ Many bug fixes and minor improvements.
```

```
Changes in v10.08:
    1/ Added internal temperature to daily and monthly summaries.
       Run Reprocess.py when upgrading from earlier versions.
    2/ Added 'prevdata' to Template.py. Allows calculations that
       compare values from different times.
    3/ Made 'pressure_offset' available to calculations in Plot.py
       and Template.py. This is only useful when using 'raw' data.
    4/ Improved synchronisation to weather station's clock when
       fetching stored data.

Changes in v10.06:
    1/ Improved localisation code.
    2/ Minor bug fixes.
    3/ Added Y axis label angle control to plots.

Changes in v10.04:
    1/ Changed version numbering to year.month.
    2/ Allowed "upload" to a local directory instead of ftp site.
    3/ Added "calc" option to text templates (Template.py).
    4/ Added -v / --verbose option to Hourly.py to allow silent operation.
    5/ Added internationalisation / localisation of some strings.
    6/ Made 'raw' data available to text templates.
    7/ Added ability to upload to Weather Underground.
    8/ Added dual axis and cumulative graph capability.

Changes in v0.9:
    1/ Added lowest daytime max and highest nighttime min temperatures
       to monthly data.
    2/ Added average temperature to daily and monthly data.
    3/ Added 'terminal' element to Plot.py templates for greater control
       over output appearance.
    4/ Added 'command' element to Plot.py templates for even more
       control, for advanced users.
    5/ Added secure upload option.
    6/ Minor speed improvements.

Changes in v0.8:
    1/ Added meteorological day end hour user preference
    2/ Attempts at Windows compatibility
    3/ Corrected decoding of wind data at speeds over 25.5 m/s
    4/ Improved speed with new data caching strategy

Changes in v0.7:
    1/ Several bug fixes, mostly around new weather stations with not
       much data
    2/ Added min & max temperature extremes to monthly data
    3/ Added template and workspace directory locations to weather.ini
    4/ Increased versatility of Plot.py with layout and title elements

Changes in v0.6:
    1/ Added monthly data
    2/ Changed 'pressure' to 'abs_pressure' or 'rel_pressure'

Changes in v0.5:
    1/ Small bug fixes.
    2/ Added start time to daily data
    3/ Replaced individual plot programs with XML "recipe" system
```

```
Changes in v0.4:
    1/ Can post brief messages to Twitter.
    2/ Now time zone aware. Uses UTC for data indexing and local time
       for graphs and text data files.

Changes in v0.3:
    1/ Now uses templates to generate text data
    2/ Added 28 day plot
    3/ Minor efficiency improvements
    4/ Improved documentation

Changes in v0.2:
    1/ Now uses Python csv library to read and write data
    2/ Creates hourly and daily summary files
    3/ Includes rain data in graphs
```

### 4.1.4 Guides utilisateur

Contenu :

#### Comment démarrer avec pyws

##### Fichiers prérequis.

- Python 2.5+ - http://python.org/ (Note : Le support de Python 3 est en développement)
- Librarie USB - option 1 (préférable pour les Mac)
  - PyUSB 1.0 - http://sourceforge.net/apps/trac/pyusb/
  - libusb 0.1 or 1.0 - http://www.libusb.org/
- Librarie USB - option 2 (Si PyUSB 1.0 n'est pas disponible)
  - PyUSB 0.4 - http://sourceforge.net/apps/trac/pyusb/
  - libusb 0.1 - http://www.libusb.org/
- Librarie USB - option 3 (préférable pour les Mac)
  - hidapi - https://github.com/signal11/hidapi
  - ctypes - http://docs.python.org/2/library/ctypes.html
- Librarie USB - option 4 :
  - hidapi - https://github.com/signal11/hidapi
  - cython-hidapi - https://github.com/gbishop/cython-hidapi
  - cython - http://cython.org/

Vous serez peut-être en mesure de les installer sur votre PC via un gestionnaire de paquet. Cette solution est beaucoup plus facile que de télécharger puis compiler les fichiers source à partir des sites de projet. Noter que certaines distributions Linux peuvent utiliser des noms différents pour certains paquets, ex. sur Ubuntu, pyusb se nomme python-usb.

En plus de ce qui précède, je recommende l'installation de pip (le paquet peut aussi se nommer python-pip) ou easy_install. Ceux-ci simplifient l'installation de logiciels par le Python Package Index (PyPI). Par exemple, PyUSB peut être installé à partir de PyPI en utilisant la commande `pip` :

```
sudo pip install pyusb
```

##### Télécharger le logiciel pyws.

Créer un dossier pour tous vos fichiers reliés à la météo et vous positionner dans ce dossier. Par exemple (avec un système d'exploitation Linux ou similaire) :

```
mkdir ~/weather
cd ~/weather
```

Vous pouvez installer pywws directement à partir de PyPI en utilisant `pip` ou `easy_install`, ou vous pouvez télécharger et extraire les fichiers dans votre répertoire météo ; ce qui a l'avantage de pouvoir facilement lire les modules Python et autres fichiers. Il vous permet également d'exécuter le logiciel pywws sans les privilèges du 'root' habituellement requis pour installer le logiciel.

**Installation facile** Voici la commande simple en une ligne :

```
sudo pip install pywws
```

Les répertoires où tout est installé dépend de votre système d'exploitation et de la version de Python employée. Les modules de pywws sont installés dans le répertoire 'site-packages' (ex. `/usr/lib/python2.7/site-packages`). Habituellement, les scripts sont installés dans `/usr/bin`, et les fichiers exemples sont installés dans `/usr/share/pywws`, mais d'autres répertoires (tel que `/usr/local/share`) peuvent être utilisés.

**Téléchargement et extraction** Vous pouvez soit télécharger une version officielle de pywws à partir de PyPI, ou vous pouvez utiliser `git` pour obtenir la plus récente version en développement.

Pour installer une version officielle, visiter http://pypi.python.org/pypi/pywws/ et télécharger le fichier .tar.gz ou .zip. Enregistrez-le dans votre dossier "weather", puis extraire tous les fichiers de l'archive, par exemple :

```
cd ~/weather
tar zxvf pywws-12.11_95babb0.tar.gz
```

ou :

```
cd ~/weather
unzip pywws-12.11_95babb0.zip
```

Ceci devrait créer un répertoire (nommé `pywws-12.11_95babb0` dans cet exemple) contenant tous les fichiers source de pywws. Il est pratique de créer un lien vers ce répertoire maladroitement nommé :

```
cd ~/weather
ln -s pywws-12.11_95babb0 pywws
```

Vous pouvez également obtenir la dernière version de développement de pywws, en utilisant `git clone`, puis utilisez `setup.py` pour compiler les fichiers de langue et la documentation :

```
cd ~/weather
git clone https://github.com/jim-easterbrook/pywws.git
cd pywws
python setup.py msgfmt
python setup.py build_sphinx
```

Après téléchargement et extraction ou le clonage du dépôt git, vous pouvez ensuite utiliser `setup.py` Pour tout construire et installer :

```
cd ~/weather/pywws
python setup.py build
sudo python setup.py install
```

Cette étape est facultative, et s'installe dans les mêmes répertoires qu'utiliserait `pip`. Si vous ne faites pas cette procédure d'installation, vous serez uniquement en mesure d'exécuter les modules pywws à partir de votre répertoire pywws.

#### (Utilisateur de Python 3 seulement) Convertir pywws pour Python 3.

Si votre version de Python par défaut est 3.x et que vous avez installé pywws à l'aide de `pip`, ou exécuté `python setup.py install`, le code aura déjà été traduit de Python 2 vers Python 3 et inclu dans le processus d'installation. Sinon, vous devrez utiliser setup.py pour faire la conversion et créer une installation de Python 3 :

```
cd ~/weather/pywws
rm -Rf build
python3 setup.py build
sudo python3 setup.py install
```

#### Tester la connexion de la station météo.

Finallement, vous êtes prêt à tester votre installation de pywws. Brancher la station météo (si ce n'est pas déjà fait), puis exécuter le module *pywws.TestWeatherStation*. Si vous avez téléchargé pywws mais ne l'avez pas installé, alors n'oubliez pas de changer de répertoire pour le dossier source de pywws. Par exemple :

```
cd ~/weather/pywws
python -m pywws.TestWeatherStation
```

Si tout fonctionne correctement, vous devriez voir apparaitre un lot de chiffres ressemblant à ceci :

```
0000 55 aa ff ff ff ff ff ff ff ff ff ff ff ff ff ff 05 20 01 51 11 00 00 00 81 00 00 0f 00 00 60 55
0020 ea 27 a0 27 00 00 00 00 00 00 00 10 10 12 13 45 41 23 c8 00 32 80 47 2d 2c 01 2c 81 5e 01 1e 80
0040 96 00 c8 80 a0 28 80 25 a0 28 80 25 03 36 00 05 6b 00 00 0a 00 f4 01 18 03 00 00 00 00 00 00 00
0060 00 00 4e 1c 63 0d 2f 01 73 00 7a 01 47 80 7a 01 47 80 e4 00 00 00 71 28 7f 25 bb 28 bd 25 eb 00
0080 0c 02 84 00 0e 01 e3 01 ab 03 dc 17 00 10 08 21 08 54 10 03 07 22 18 10 08 11 08 30 10 04 21 16
00a0 26 08 07 24 17 17 08 11 01 06 10 09 06 30 14 29 09 01 06 07 46 09 06 30 14 29 09 01 06 07 46 08
00c0 08 31 14 30 10 05 14 15 27 10 01 26 20 47 09 01 23 05 13 10 01 26 20 47 09 01 23 05 13 10 02 22
00e0 11 06 10 02 22 11 06 08 07 07 19 32 08 12 13 22 32 08 09 07 08 48 01 12 05 04 43 10 02 22 14 43
```

Si ce test ne fonctionne pas, plusieurs facteurs peuvent être en cause, mais le plus courant est un problème de 'permissions' ; ce qui peut être vérifié en exécutant la commande suivante avec les droits 'root' :

```
sudo python -m pywws.TestWeatherStation
```

Si celà fonctionne, vous devriez être capable de permettre à votre compte utilisateur normal d'accéder à votre station météo en créant une règle 'udev'. Voir la page de compatibilité du wiki http://code.google.com/p/pywws/wiki/Compatibility (en anglais) pour plus de détails.

Pour tout autre problème, n'hésitez pas à demander de l'aide sur la liste de publipostage de pywws : http://groups.google.com/group/pywws

#### Configurer votre station météo.

Si vous ne l'avez pas déjà fait, configurez votre station météo pour afficher la bonne pression atmosphérique relative. (Voir le manuel pour plus d'informations sur la façon de faire.) pywws obtient le décalage entre la pression relative et absolue de la station, donc cela doit être défini avant d'utiliser pywws .

Vous pouvez obtenir la bonne pression relative de votre emplacement en recherchant sur Internet des rapports météorologiques d'une station à proximité, idéalement une source officielle, tel un aéroport. Il est préférable de le faire durant une période calme lorsque la pression est presque constante sur une longue période.

Si vous modifiez le décalage, à tout moment, vous pouvez mettre à jour toutes vos données stockées en exécutant *pywws.Reprocess*.

#### Configurer l'intervalle d'enregistrement de la station météo.

Votre station météo a probablement quitté l'usine avec un intervalle d'enregistrement de 30 minutes. Ceci permet à la station d'enregistrer environ 11 semaines de données. La plupart des utilisateurs de pywws configurent leur ordinateur pour lire les données de la station à chaque heure, ou plus fréquemment et souhaitent que la station contienne assez de données pour couvrir d'éventuelles panne d'ordinateur. L'intervalle recommandé est de 5 minutes, ce qui représente 2 semaines de données. Utilisez le programme *pywws.SetWeatherStation* pour fixer l'intervalle :

```
python -m pywws.SetWeatherStation -r 5
```

#### Enregistrer les données de votre station météo.

En premier lieu, choisissez un dossier pour entreposer toute les données de votre station météo. Ces données étant écrites très fréquemment, un disque rigide sera préférable à une mémoire Flash (USB), puisque ces dernières ont un nombre d'écriture limité. Dans la plupart des cas, le répertoire personnel est préférable, par exemple :

```
mkdir ~/weather/data
```

Ce répertoire est référé dans la documentation pywws en tant que votre 'répertoire de données'.

Assurez-vous que votre ordinateur ait la date et l'heure précise, ainsi que le fuseau horaire approprié, puisqu'ils seront utilisés pour identifier les données de votre station météo. Si vous ne l'avez pas déjà fait, il pourrait être utile d'utiliser la synchronisation horaires par réseau (NTP) pour synchroniser votre ordinateur à un 'serveur de temps'.

La première fois que vous exécutez *pywws.LogData* un fichier de configuration, nommé 'weather.ini', sera créé dans votre répertoire de données, puis s'arrêtera. Vous devez éditer ce fichier de configuration et y modifier la ligne `ws type = Unknown` à `ws type = 1080` ou `ws type = 3080`. (Si la console de votre station météo affiche la luminosité solaire, vous avez une station de type 3080, sinon, vous indiquez le type 1080.) Puis exécutez *pywws.LogData* de nouveau. Cette fois, l'exécution peut durer quelques minutes, puisqu'il copiera l'intégralité des données météo emmagasinées dans la mémoire de votre station météo. Le programme *pywws.LogData* possède une option 'verbose' qui augmente le nombre de messages affichés pendant l'exécution. Cette option est utile lors d'exécution manuelle, par exemple :

```
python -m pywws.LogData -vvv ~/weather/data
```

(Remplacez `~/weather/data` par votre répertoire de données, si différent.)

Vous devriez maintenant avoir quelques fichiers de données à regarder. Par exemple :

```
more ~/weather/data/weather/raw/2012/2012-12/2012-12-16.txt
```

(Remplacez année, mois et jour par la date pour laquelle vous devriez avoir des données.)

#### Convertir les anciennes données de EasyWeather (optionnel).

Si vous utilisiez EasyWeather avant de décider d'utiliser pywws, vous pouvez convertir les données que EasyWeather a enregistré, vers format de pywws. Localisez votre fichier EasyWeather.dat et convertissez-le ainsi :

```
python -m pywws.EWtoPy EasyWeather.dat ~/weather/data
```

#### Indiquer certaines options de configuration.

Après avoir exécuté *pywws.LogData*, il devrait y avoir un fichier de configuration nommé 'weather.ini' dans votre répertoire de données. Ouvrez ce fichier avec un éditeur de texte. Vous devriez y trouver des lignes semblables à celle-ci :

```
[config]
ws type = 1080
logdata sync = 1
```

Vous devez ajouter une nouvelle entrée nommée `day end hour` dans la section `[config]`. Ceci indique à pywws quelle convention vous souhaitez utiliser pour le calcul du sommaire des données quotidiennes. En Grande-Bretagne, le 'jour météorologique' s'étale habituellement de 09 :00 à 09 :00 GMT (10 :00 à 10 :00 BST durant l'été), j'utilise donc 9 pour l'heure de fin du jour. Dans d'autres pays une valeur de 24 (ou 0) peut être souhaitable. Noter que cette valeur est définie à l'heure hivernale locale. Vous ne devriez pas avoir besoin de changer cette valeur pendant la saison chaude.

Après édition, votre fichier weather.ini devrait resembler à ceci :

```
[config]
ws type = 1080
logdata sync = 1
day end hour = 9
```

Pour plus de détails sur les options du fichier de configuration, voir weather.ini - format du fichier de configuration.

#### Traitement des données brutes.

*pywws.LogData* ne fait que copier les données brutes à partir de la station météo. Pour faire quelque chose d'utile avec ces données vous avez probablement besoin des sommaires horaire, quotidien et mensuel. Ces sommaires sont produits à partir du programme *pywws.Process*. Par exemple :

```
python -m pywws.Process ~/weather/data
```

Vous devriez avoir ainsi quelques fichiers traités à consulter :

```
more ~/weather/data/weather/daily/2012/2012-12-16.txt
```

Si vous changez votre période de fin du jour avec l'option de configuration `day end hour`, vous devrez re-traiter toutes vos données météo. Pour ce faire, exécutez le programme *pywws.Reprocess* :

```
python -m pywws.Reprocess ~/weather/data
```

Vous êtes maintenant prêt pour fixer une journalisation régulière ou continue, tel que décrit dans la section Comment configurer la journalisation horaire avec pywws ou Comment configurer le mode 'live' avec pywws.

#### Lire la documentation.

Le dossier doc du répertoire source de pywws contient une version HTML et une version en texte clair de la documentation ( à moins que vous ayez fait une installation directe avec `pip`). Les fichiers HTML peuvent être lus avec tous navigateurs. Débutez avec l'index (pywws) et suivez les liens à partir de ce point.

Commentaires ou questions ? SVP, souscrivez à la liste d'envoi de pywws http://groups.google.com/group/pywws et laissez-le nous savoir.

## Comment configurer la journalisation horaire avec pywws

### Introduction

Pywws offre deux modes de fonctionnement très différents. Habituellement le programme *Hourly* devrait être exécuté à intervale régulier (habituellement chaque heure) à partir d'une tâche programmée (Cron). Ceci convient aux sites Web plutôt statiques, mais des mises à jour plus fréquentes peuvent être utiles pour des sites comme Weather Underground (http://www.wunderground.com/). Le plus récent programme *LiveLog* s'exécute continuellement et peut envoyer des données à chaque 48 secondes.

Notez : bien que ce document (et le nom du programme) réfère à la journalisation horaire (hourly), vous pouvez exécuter le programme *Hourly* aussi souvent ou peu fréquemment que vous le souhaitez, mais n'essayez pas de l'exécuter plus souvent que le double de la fréquence d'historique. Par exemple, si votre intervalle d'historique est de 10 minutes, n'exécutez pas *Hourly* plus fréquemment qu'aux 20 minutes.

### Mise en route

Avant tout, vous devez installer pywws et vous assurer qu'il reçoit bien les informations de votre station météo. Voir Comment démarrer avec pywws pour plus de détails.

Essayez d'exécuter *Hourly* à partir de la ligne de commande, avec un haut niveau de commentaires pour que vous puissiez voir ce qui se passe :

```
python -m pywws.Hourly -vvv ~/weather/data
```

En moins de cinq minutes (assumant que votre intervalle de relevé soit de 5 minutes) vous devriez voir le message 'live_data new ptr', suivi par la recherche et le traitement de nouvelles données de la station météorologique.

### Configurer l'emplacement des fichiers

Ouvrez votre fichier weather.ini avec un éditeur de texte. Vous devriez avoir une section `[paths]` similaire à ce qui suit (où `xxx` est votre nom d'usager) :

```
[paths]
work = /tmp/weather
templates = /home/xxx/weather/templates/
graph_templates = /home/xxx/weather/graph_templates/
```

Éditez pour correspondre à votre installation et à vos préférences. `work` est un dossier temporaire utilisé pour emmagasiner les fichiers intermédiaires, `templates` est le dossier où vous gardez vos fichiers de gabarit texte et `graph_templates` est le dossier où vous gardez vos fichiers de gabarit graphes. Ne pas utiliser les dossiers exemple de pywws pour celà, puisqu'ils seront écrasés lors de mise à jour de pywws.

Copiez vos fichiers de gabarits texte et graphes dans les dossiers appropriés. Vous pouvez trouver les quelques exemples fournis avec pywws fort utiles pour débuter. Si vous avez installé pywws avec la commande `pip`, les exemples devrait se trouver dans le dossier `/usr/share/pywws` ou `/usr/local/share/pywws` ou similaire.

### Configurer les tâches périodiques

Dans weather.ini vous devriez avoir les sections `[logged]`, `[hourly]`, `[12 hourly]` et `[daily]` similaires à celle-ci :

```
[logged]
services = []
twitter = []
plot = []
text = []

[hourly]
...
```

Ces sections spécifient ce que devrait faire *Hourly* à chaque exécution. Les tâches dans la section `[logged]` sont réalisées à chaque fois qu'une nouvelle donnée est enregistrée, les tâches dans la section `[hourly]` sont exécutées à chaque heure, les tâches dans la section `[12 hourly]` sont lancées deux fois par jour, tandis que les tâches dans la section `[daily]` sont réalisées une fois par jour.

Les entrées dans `services` sont une liste de services météo en ligne sur lesquels envoyer vos données météo. Les entrées `plot` et `text` sont des listes de fichiers gabarits graphes et textes à téléverser sur votre site web, et l'entrée `twitter` est une liste de gabarits pour les messages à poster sur Twitter. Ajoutez le nom de vos fichiers de gabarit et de services météo à l'entrée correspondante, par exemple :

```
[logged]
services = ['underground', 'metoffice']
twitter = []
plot = []
text = []

[hourly]
services = []
twitter = ['tweet.txt']
plot = ['7days.png.xml', '24hrs.png.xml', 'rose_24hrs.png.xml']
text = ['24hrs.txt', '6hrs.txt', '7days.txt']

[12 hourly]
services = []
twitter = []
plot = []
text = []

[daily]
services = []
twitter = ['forecast.txt']
plot = ['28days.png.xml']
text = ['allmonths.txt']
```

Vous pouvez réaliser un test de bon fonctionnement en enlevant la section `last update` du fichier weather.ini, puis exécutez *Hourly* de nouveau :

```
python -m pywws.Hourly -v ~/weather/data
```

Modifié dans la version 13.05_r1009 : The last update information was previously stored in weather.ini, with `last update` entries in several sections.

#### Utilisation d'un script

L'installation de pywws inclu un petit script `pywws-hourly.py` à installer dans `/usr/bin` ou `/usr/local/bin` ou similaire. Vous devriez être en mesure d'utiliser ce script en exécutant *Hourly* :

```
pywws-hourly.py -v ~/weather/data
```

#### Exécuter en tant que tâche 'cron'

La plupart des systèmes UNIX/Linux possèdent un service 'cron' qui à pour but d'exécuter un programme spécifié à un intervalle défini, même si vous n'êtes pas connecté à votre session. Vous éditez un fichier 'crontab' pour spécifier ce qui doit être exécuté et quand. Par exemple, pour exécuter *Hourly* à chaque heure, à zéro minutes de l'heure :

```
0 * * * *       pywws-hourly.py /home/jim/weather/data
```

Ceci devrait fonctionner, mais si ce n'est pas le cas vous n'obtiendrez probablement pas de messages d'erreur vous indiquant ce qui ne va pas. Il est préférable d'exécuter un script qui lance *Hourly* puis vous envoi un courriel avec tout message qui aurait pu être produit. Voici le script que j'utilise :

```
#!/bin/sh
#
# weather station logger calling script

if [ ! -d /data/weather/ ]; then
  exit
  fi

log=/var/log/log-weather

pywws-hourly.py -v /data/weather >$log 2>&1

# mail the log file
/home/jim/scripts/email-log.sh $log "weather log"
```

Vous devrez éditer beaucoup pour adapter à vos emplacements de dossier et ainsi de suite, mais il donne une certaine idée de ce qui peut être fait.

Commentaires ou questions ? SVP, souscrivez à la liste d'envoi de pywws http://groups.google.com/group/pywws et laissez-le nous savoir.

### Comment configurer le mode 'live' avec pywws

#### Introduction

Pywws offre deux modes de fonctionnement très différents. Habituellement le programme *Hourly* devrait être exécuté à intervale régulier (habituellement chaque heure) à partir d'une tâche programmée (Cron). Ceci convient aux sites Web plutôt statiques, mais des mises à jour plus fréquentes peuvent être utiles pour des sites comme Weather Underground (http://www.wunderground.com/). Le plus récent programme *LiveLog* s'exécute continuellement et peut envoyer des données à chaque 48 secondes.

#### Mise en route

Avant tout, vous devez installer pywws et vous assurer qu'il reçoit bien les informations de votre station météo. Voir Comment démarrer avec pywws pour plus de détails.

Essayez d'exécuter *LiveLog* à partir de la ligne de commande, avec un haut niveau de commentaires pour que vous puissiez voir ce qui se passe :

```
python -m pywws.LiveLog -vvv ~/weather/data
```

En moins de cinq minutes (assumant que vous avez un intervalle de relevé de 5 minutes) vous devriez voir le message 'live_data new ptr', suivi par la recherche et le traitement de nouvelles données de la station météorologique. Laissez *LiveLog* fonctionner une ou deux minutes de plus, puis arrêtez le processus en tapant '<Ctrl>C'.

#### Configurer l'emplacement des fichiers

Ouvrez votre fichier weather.ini avec un éditeur de texte. Vous devriez avoir une section `[paths]` similaire à ce qui suit (où `xxx` est votre nom d'usager) :

```
[paths]
work = /tmp/weather
templates = /home/xxx/weather/templates/
graph_templates = /home/xxx/weather/graph_templates/
```

Éditez pour correspondre à votre installation et à vos préférences. `work` est un dossier temporaire utilisé pour emmagasiner les fichiers intermédiaires, `templates` est le dossier où vous gardez vos fichiers de gabarit texte et `graph_templates` est le dossier où vous gardez vos fichiers de gabarit graphes. Ne pas utiliser les dossiers exemple de pywws pour celà, puisqu'ils seront écrasés lors de mise à jour de pywws.

Copiez vos fichiers de gabarits texte et graphes dans les dossiers appropriés. Vous pouvez trouver les quelques exemples fournis avec pywws fort utiles pour débuter. Si vous avez installé pywws avec la commande `pip`, les exemples devrait se trouver dans le dossier `/usr/share/pywws` ou `/usr/local/share/pywws` ou similaire.

#### Configurer les tâches périodiques

Dans weather.ini vous devriez avoir une section `[live]` similaire à celle-ci :

```
[live]
services = []
twitter = []
plot = []
text = []
```

Cette section spécifie ce que devrait faire pywws à chaque fois qu'il reçoit une lecture de la station météo, ex. toutes les 48 secondes. Les entrées `services` sont des listes de services météo en ligne sur lesquels envoyer vos données météo, ex. `['underground']`. Les entrées `plot` et `text` sont des listes de fichiers gabarits de graphe et de texte à téléverser sur votre site web, et l'entrée `twitter` est une liste de gabarits pour les messages à poster sur Twitter. Vous devriez probablement laisser toutes ces entrées vides, sauf pour `services`.

Si vous utilisez YoWindow (http://yowindow.com/) vous pouvez ajouter l'entrée à la section `[live]` pour spécifier votre fichier YoWindow, ex. :

```
[live]
yowindow = /home/jim/data/yowindow.xml
services = ['underground']
...
```

Si vous ne les avez pas déjà, créez quatre sections supplémentaires dans votre fichier weather.ini : `[logged]`, `[hourly]`, `[12 hourly]` et `[daily]`. Ces sections doivent avoir des entrées similaires à la section `[live]`, et spécifiez ce qui doit être fait chaque fois qu'une donnée est enregistrée (5 à 30 minutes, dépendant de votre intervalle), chaque heure, deux fois par jour et chaque jour. Ajoutez les noms de vos fichiers de gabarit à l'entrée appropriée, par exemple :

```
[logged]
services = ['underground', 'metoffice']
twitter = []
plot = []
text = []

[hourly]
services = []
twitter = ['tweet.txt']
plot = ['7days.png.xml', '24hrs.png.xml', 'rose_24hrs.png.xml']
text = ['24hrs.txt', '6hrs.txt', '7days.txt']

[12 hourly]
services = []
twitter = []
plot = []
text = []

[daily]
services = []
twitter = ['forecast.txt']
plot = ['28days.png.xml']
text = ['allmonths.txt']
```

#### Utilisation d'un script

L'installation de pywws inclu un petit script `pywws-livelog.py` qui se trouve dans le dossier `/usr/bin` ou `/usr/local/bin` ou similaire. Vous devriez être en mesure d'utiliser ce script pour exécuter *LiveLog* :

```
pywws-livelog.py -v ~/weather/data
```

#### Exécuter en arrière-plan

Afin que *LiveLog* continue de fonctionner après avoir fini d'utiliser votre ordinateur, vous devez exécuter ce programme en tant que 'tâche en arrière-plan'. Sur la plupart des systèmes Linux / UNIX vous pouvez faire celà en ajoutant simplement le symbole ('&') à la fin de la ligne de commande. Par exemple :

```
pywws-livelog.py ~/weather/data &
```

Cependant, il peut être utile de savoir ce qui ne va pas si le programme se bloque pour une raison quelconque. *LiveLog* peut enregistrer ces messages dans un fichier d'historique, pour ce faire spécifiez l'option `-l` :

```
pywws-livelog.py -v -l ~/weather/data/pywws.log ~/weather/data &
```

#### Redémarrage automatique

Il y a une plusieurs manières de configurer un système Linux pour lancer un programme au démarrage de l'ordinateur. Habituellement pour ce faire, vous devez mettre un fichier dans `/etc/init.d/`, ce qui requière les privilèges 'root'. Un problème un peu plus difficile est d'assurer qu'un programme redémarre s'il se bloque. Ma solution à ces deux problèmes consiste à exécuter le script cron suivant, toutes les heures.

```
#!/bin/sh

pidfile=/var/run/pywws.pid
datadir=/data/weather
logfile=$datadir/live_logger.log

# exit if process is running
[ -f $pidfile ] && kill -0 `cat $pidfile` && exit

# email last few lines of the logfile to see why it died
if [ -f $logfile ]; then
  log=/var/log/log-weather
  tail -40 $logfile >$log
  /home/jim/scripts/email-log.sh $log "weather log"
  rm $log
  fi

# restart process
pywws-livelog.py -v -l $logfile $datadir &
echo $! >$pidfile
```

Ce script enregistre l'identificateur de processus exécutant *LiveLog* dans le pidfile. Si le processus fonctionne, le script ne fait rien. Si le processus a bloqué, il m'envoie un courriel contenant les 40 dernières lignes du fichier d'historique (en utilisant un script qui crée un message et l'envoi par sendmail) puis redémarre *LiveLog*. Vous devrez éditer beaucoup pour adapter à vos emplacements de dossier et ainsi de suite, mais il donne une certaine idée de ce qui doit être fait.

Commentaires ou questions ? SVP, souscrivez à la liste d'envoi de pywws http://groups.google.com/group/pywws et laissez-le nous savoir.

### Comment intégrer divers services météorologiques à pywws

Ce guide donne de brèves instructions sur comment utiliser pywws avec d'autres services météorologiques et logiciels. Il n'est pas exhaustive, et certains services (comme Twitter) sont couverts en détail ailleurs.

#### YoWindow

YoWindow est un widget météo qui peut afficher des données provenant d'une source internet ou de votre station météo. Pour afficher les données de votre station, pywws a besoin d'écrire dans un fichier local, en général toutes les 48 secondes lorsque de nouvelles données sont reçues. C'est facile à faire :

1. Arrêter toute instance de pywws
2. Copier le gabarit exemple `yowindow.xml` dans votre répertoire de gabarits text.
3. Si ce n'est déjà fait, éditez le fichier `weather.ini` et configurez la clé `local_files` dans la section `[paths]` pour un répertoire approprié pour votre fichier yowindow.
4. Ajoutez le modèle yowindow dans les tâches `[live]` de `weather.ini`. Mettre son paramètre à `'L'` si le résultat est copié dans votre répertoire local au lieu d'être téléchargés sur un site ftp :

```
    [live]
    text = [('yowindow.xml', 'L')]
```

5. Redémarrez l'enregistrement live de pywws.

Vous pouvez vérifier que le fichier est mis à jour toutes les 48 secondes en utilisant `more` ou `cat` pour afficher à l'écran.

Finallement, configurez yowindow pour l'utilisation de ce fichier. Voir http://yowindow.com/pws_setup.php pour les instructions pour ce faire.

#### Twitter

Voir Comment configurer pywws pour poster des messages sur Twitter pour les instructions détaillées.

#### Weather Underground

Weather Underground (ou Wunderground) est l'un des plus anciens sites météorologiques dans le monde. Comme pour beaucoup d'autres services, pywws peut leur envoyer vos données météo via l'internet. Le module *pywws.toservice* gère cette communication pour toute une gamme de services en ligne .

La première étape consiste à vous créer un compte Weather Underground à cette adresse http://www.wunderground.com/members/signup.asp. Puis utilisez le formulaire "Add A Station" pour fournir les détails de votre station tel que la localisation et son type. Vous devriez par la suite obtenir un identificateur de station (station ID) et un mot de passe – prenez ces informations en note.

Assurez vous qu'aucune autre instance de pywws ne fonctionne, puis exécutez le programme *pywws.toservice* directement :

```
python -m pywws.toservice ~/weather/data underground
```

Celà devrait échouer, comme vous n'avez pas encore défini l'ID de votre station ou le mot de passe, mais ceci crée les entrées nécessaires dans le fichier `weather.ini` pour que vous puissiez l'éditer. Ouvrez `weather.ini` et trouvez la section `[underground]` :

```
[underground]
station = unknown
password = unknown
```

Remplacez les valeures `unknown` par votre Station ID et votre mot de passe.

Maintenant, essayez *pywws.toservice* de nouveau :

```
python -m pywws.toservice ~/weather/data underground
```

Si cela a fonctionné, vous pouvez télécharger votre dernier 7 jours de données. Notez que cela peut prendre un temps assez long, surtout si vous avez un intervalle d'enregistrement court. En premier éditez `status.ini` et supprimez l'entrée `underground` de la section `[last update]`. Ensuite, exécutez *pywws.toservice* avec l'option 'catchup' (rattrapage) et haute verbosité afin que vous puissiez voir le fonctionnement :

```
python -m pywws.toservice -vvc ~/weather/data underground
```

Lorsque tout fonctionne, vous pouvez ajouter 'underground' à la section des tâches `[logged]` du fichier `weather.ini` :

```
[logged]
services = ['underground']
```

**Mise à jour "RapidFire"** Weather Underground a une seconde URL de téléversement pour les mises à jour en temps réel à un interval d'aussi peu que 2.5 secondes. Si vous exécutez pywws en mode 'live logging' (voir Comment configurer le mode 'live' avec pywws) vous pouvez l'utiliser pour envoyez des mises à jour aux 48 secondes, en ajoutant 'underground_rf' à la section des tâches `[live]` du fichier `weather.ini` :

```
[live]
services = ['underground_rf']
```

On ne sait ne pas si Weather Underground approuve l'envoi de mises à jour RapidFire et normales pour la même station. (Voir http://wiki.wunderground.com/index.php/PWS_-_Upload_Protocol#RapidFire_Updates.) Si vous utilisez seulement RapidFire, il y a possibilité de trouées dans l'historique de votre station, si elle devient "hors onde" pour une raison quelconque.

Commentaires ou questions ? SVP, souscrivez à la liste d'envoi de pywws http://groups.google.com/group/pywws et laissez-le nous savoir.

### Comment configurer pywws pour poster des messages sur Twitter

#### Fichiers prérequis.

Postersur Twitter requière certains logiciels supplémentaires. Voir Pré-requis - *Mise à jour Twitter*.

#### Créer un compte Twitter.

Vous pouvez poster des mise à jour météo sur votre compte Twitter 'normal', mais je crois qu'il est préférable d'avoir un compte distinct réservé à vos relevés météo. Ce qui pourrait être fort utile à une personne vivant dans votre voisinage, mais ne souhaitant pas savoir ce que vous avez pris au petit déjeuner.

#### Autoriser pywws à poster sur votre compte Twitter.

Si vous exécutez pywws sur un appareil de faible puissance tel qu'un routeur, il peut plus simple d'exécuter cette étape d'autorisation sur un autre ordinateur en autant que `python-oauth2` y est installé. Utiliser un répertoire 'data' vide – un fichier `weather.ini` sera créé, dont le contenu pourra être copié dans votre fichier `weather.ini` réel en utilisant tout éditeur de texte.

Assurez vous qu'aucune autre instance de pywws ne fonctionne, puis exécutez le module *`TwitterAuth`* :

```
python -m pywws.TwitterAuth /data/weather
```

(Remplacez `/data/weather` par votre répertoire de données.)

Ceci ouvrira une fenêtre de navigation (ou vous donnera une URL à copier dans votre navigateur) où vous pourrez vous connecter à votre compte Twitter et autoriser pywws à poster. Votre navigateur vous affichera alors un nombre à 7 chiffre que vous aurez besoin de copier dans le programme *`TwitterAuth`*. Si tout s'est bien déroulé, votre fichier `weather.ini` devrait comporter une nouvelle section `[twitter]` avec les entrées `secret` et `key`. (Ne révéler à quiconque.)

#### Ajouter vos données de localisation (optionel).

Éditez votre fichier `weather.ini` et ajoutez les entrées `latitude` et `longitude` à votre section `[twitter]`. Par exemple :

```
[twitter]
secret = xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
key = xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
latitude = 51.501
longitude = -0.142
```

#### Créer un gabarit.

Les messages pour Twitter sont générés en utilisant un gabarit, tout comme sont créés les fichiers à téléverser sur votre site web. Copiez le gabarit exemple 'tweet.txt' dans votre dossier de gabarit, puis testez-le :

```
python -m pywws.Template /data/weather ~/weather/templates/tweet.txt tweet.txt
cat tweet.txt
```

(Remplacez `/data/weather` et `~/weather/templates` par vos répertoires de données et de gabarits.) Si vous avez besoin de modifier le gabarit (ex. pour changer les unités de mesure ou la langue utilisée) vous pouvez l'éditer maintenant ou ultérieurement.

#### Poster votre premier Tweet météo.

Maintenant tout est pret pour exécuter *`ToTwitter`* :

```
python -m pywws.ToTwitter /data/weather tweet.txt
```

Si celà fonctionne, Votre nouveau compte Twitter aura posté son premier relevé météo. (Vous devriez supprimer le fichier tweet.txt maintenant.)

#### Ajouter les envois sur Twitter à vos tâche horaire.

Éditez votre fichier `weather.ini` et modifiez la section `[hourly]`. Par exemple :

```
[hourly]
services = []
plot = ['7days.png.xml', '24hrs.png.xml', 'rose_12hrs.png.xml']
text = [('tweet.txt', 'T'), '24hrs.txt', '6hrs.txt', '7days.txt']
```

Notez l'usage de l'indicateur `'T'` – ce qui indique à pywws de `'Tweeter'` le résultat du gabarit au lieu de le verser sur votre site par ftp.

Vous pourriez tout aussi bien choisir de plutôt modifier les sections `[logged]`, `[12 hourly]` ou `[daily]`, mais je crois que `[hourly]` est le plus approprié pour les envois sur Twitter.

Modifié dans la version 13.06_r1015 : added the `'T'` flag. Previously Twitter templates were listed separately in `twitter` entries in the `[hourly]` and other sections. The older syntax still works, but is deprecated.

Commentaires ou questions ? SVP, souscrivez à la liste d'envoi de pywws http://groups.google.com/group/pywws et laissez-le nous savoir.

## Comment utiliser pywws dans une autre langue

### Introduction

Certaines parties de pywws peuvent être configurées pour utiliser votre langue locale au lieu de l'Anglais Britannique. Ceci requière un fichier de langage contenant la traduction des différentes phrases utilisées par pywws. Le projet repose sur les utilisateur qui fournissent ces traductions. Ce document décrit comment créer un fichier de langage.

La documentation de pywws peut aussi être traduite dans d'autres langues. C'est beaucoup de travail, mais pourrait être très utile aux utilisateurs potentiels qui ne lisent pas très bien l'anglais.

### Prérequis.

En plus du logiciel pywws vous devez installer le paquet utilitaire GNU d'internationalisation `gettext`. Celui-ci est disponible dans la plupart des dépôts standard des distributions Linux, ou vous pouvez le télécharger à partir de http://www.gnu.org/software/gettext/ si vous devez le compiler vous même.

### Choisir votre code de langue.

Les ordinateur utilisent les étiquettes d'identification de langues IETF (voir http://fr.wikipedia.org/wiki/%C3%89tiquette_d%27identification_de_langues_IETF). Par exemple, la Grande-Bretagne utilise `en_GB` qui est composé de 2 parties : `en` pour Anglais, et `GB` la version Britanique. Pour trouver le code correspondant à votre langue, consultez la liste sur http://www.gnu.org/software/gettext/manual/gettext.html#Language-Codes.

### Mise en route

Votre dossier pywws doit déjà contenir un sous-dossier nommé translations. Celui-ci contient l'ensemble des fichiers de langue actuel, par exemple `translations/fr/pywws.po` contient la traduction Française. Si une de ces langues est ce que vous désirez, alors éditez votre fichier weather.ini et ajoutez une entrée `language` à la section `[config]`, par exemple :

```
[config]
day end hour = 21
gnuplot encoding = iso_8859_1
language = sv
```

Vous devrez toujours compiler et installer le fichier de langue choisie. Ce qui est fait avec `setup.py` :

```
python setup.py msgfmt
```

S'il n'y a pas déjà de fichier pour votre langage, le reste de ce document vous indique comment en créer un.

### Créer un fichier de langue.

La première étape consiste à créer un fichier contenant les phrases que vous devez traduire. Par exemple, pour créer un fichier source pour la langue française (code `fr`) :

```
python setup.py xgettext
python setup.py msgmerge --lang=fr
```

Ce qui vous demandera de confirmer votre adresse courriel, puis créera un fichier `pywws.po` dans le répertoire `translations/fr`. Vous devez maintenant éditer le fichier `pywws.po` et remplir chaque ligne `msgstr` avec une traduction de la ligne `msgid` immédiatement au-dessus. La raison d'inclure votre adresse courriel est de permettre à chacun ayant une question à propos de la traduction de vous rejoindre. Sentez vous libre de mettre une adresse invalide si vous êtes préoccupé par le maintien de votre vie privée.

Après avoir édité le nouveau fichier de langue, il doit être compilé pour que pywws puisse l'utiliser. Ce qui est fait avec la commande `msgfmt` :

```
python setup.py msgfmt
```

N'oubliez pas de faire celà à chaque fois que vous éditez un fichier de langue.

#### Tester la traduction de pywws.

Le module *Localisation* peut être utilisé pour faire un test rapide de l'installation de votre fichier de langue :

```
python -m pywws.Localisation -t fr
```

Cela devrait afficher quelque chose comme ceci :

```
Locale changed from (None, None) to ('fr_FR', 'UTF-8')
Translation set OK
Locale
  decimal point: 23,2
  date & time: lundi, 17 décembre (17/12/2012 16:00:48)
Translations
  'NNW' => 'NNO'
  'rising very rapidly' => 'en hausse très rapide'
  'Rain at times, very unsettled' => 'Quelques précipitations, très perturbé'
```

Éditez l'entrée de langue dans votre fichier `weather.ini` pour utiliser votre code de langue (ex. `fr`), puis essayer d'utiliser *Plot* pour dessiner un graphe. L'axe des X du graphe devrait maintenant être étiquetée dans votre langue, utilisant la traduction que vous avez fournis pour 'Time', 'Day' ou 'Date'.

#### Traduire la documentation.

Le système utilisé pour convertir les phrases utilisées dans pywws peut également servir à traduire la documentation. La commande pour extraire les phrases de la documentation est très similaire :

```
python setup.py xgettext_doc
```

Notez que ceci requière le logiciel sphinx utilisé pour 'compiler' la documentation. Après extraction des phraes, il créé les fichiers source pour votre langue. Dans cet exemple la langue est Français, avec le code sur deux lettres `fr` :

```
python setup.py msgmerge --lang=fr
```

Ceci créé quatre fichiers (`index.po`, `essential.po`, `guides.po` et `api.po`) lesquels contiennent les textes (souvent des paragraphes entier) extraits à partir des différentes parties de la documentation.

Ces fichiers peuvent être édités de la même manière que le fichier `pywws.po`. Complétez chaque `msgstr` avec une traduction de la phrase `msgid` juste au-dessus. Notez que certaines chaînes (par exemple, URL et liens vers d'autres parties de la documentation) ne doivent pas être traduites. Dans ces cas, le `msgstr` doit être laissé vide.

Traduire toute la documentation de pywws représente beaucoup de travail. Toutefois, lorsque la documentation est compilée toutes les chaînes non traduites reprennent leur valeur anglaise originale. Cela signifie qu'une traduction

partielle pourrait quand même être utile - Je suggère de commencer par la première page de la documentation, `index.po`.

#### Visionner votre documentation traduite.

Tout d'abord convertir les fichiers de langue nouvellement édités :

```
python setup.py msgfmt
```

Puis. effacez l'ancienne documentation (si existante) et refaire en utilisant votre langage :

```
rm -Rf doc/html/fr
LANG=fr python setup.py build_sphinx
```

Notez que la commande `build_sphinx` n'a pas d'option `--lang`, donc le la langue est définie par une variable d'environnement temporaire.

Finallement, vous pouvez voir la documentation traduite en utilisant un navigateur web pour afficher le fichier `doc/html/fr/index.html`.

#### Mettre à jour les fichiers de langue.

Comme pywws évolue, de nouvelles phrases peuvent avoir été ajoutées et nécessiteront que votre fichier de traduction évolu lui aussi ; ce qui est très facile à faire. Premièrement, Vous devez ré-extraire les phrases à traduire, puis les fusionner avec les fichier de langue existant. Ce qui est fait en répétant les commandes utilisées pour créer les fichiers :

```
python setup.py xgettext
python setup.py xgettext_doc
python setup.py msgmerge --lang=fr
```

Ceci devrait ajouter de nouvelles phrases à vos fichiers de langue, sans modifier les phrases que vous avez déjà traduits.

Si le source original Anglais a changé depuis votre dernière traduction, certaines phrases peuvent être marquées par gettext comme `#, fuzzy` (floue) . Vous devez vérifier que votre traduction est toujours adéquate pour ces phrases – Le changement peut être mineur (par ex. une correction d'orthographe) mais il pourrait être très important. Lorsque vous avez vérifié (et corrigée si nécessaire) la traduction, enlever la ligne `#, fuzzy`.

#### Envoyer la traduction à Jim

I'm sure you would like others to benefit from the work you've done in translating pywws. Please, please, please send a copy of your language file(s) (for example `pywws.po`) to jim@jim-easterbrook.me.uk. When you send a new translation you need to include details of which pywws version it is based on – the easiest way to do this is to include the value of `commit` from the file `pywws/version.py` in your email.

Commentaires ou questions ? SVP, souscrivez à la liste d'envoi de pywws http://groups.google.com/group/pywws et laissez-le nous savoir.

### weather.ini - format du fichier de configuration

Presque toute la configuration de pywws est fait via un seul fichier dans le répertoire de donnée : weather.ini. Ce fichier a été structuré de manière similaire aux fichiers INI de Microsoft Windows. Il est divisé en 'sections', qui contiennent un nom d'entrée 'nom = valeur'. L'ordre d'apparition des sections n'a pas d'importance.

Tout éditeur de texte brut peut être utilisé pour éditer le fichier. (Ne pas essayer d'éditer le fichier pendant le fonctionnement de pywws.) Dans plusieurs cas, pywws initialisera les entrées avec des valeurs importantes.

Un autre fichier, status.ini, est utilisé pour emmagasiner certaines informations que pywws utilise à l'interne. Il est décrit à la fin de ce document. En utilisation normale, vous ne devriez pas avoir à éditer ce fichier.

Les sections suivantes sont présentement utilisées :

- config : diverses configuration système.
- paths : dossiers dans lesquels sont emmagasinés les gabarits, etc.
- live : tâches à accomplir à chaque 48 secondes.
- logged : tâches à accomplir chaque fois que la station enregistre une lecture.
- hourly : tâches à réaliser à chaque heure.
- 12 hourly : tâches à réaliser à chaque 12 heures.
- daily : tâches à faire chaque jour.
- ftp : configuration pour le téléversement du un site web.
- twitter : configuration pour poster sur Twitter.
- underground, metoffice, temperaturnu etc : configuration pour poster sur ces 'services'.

#### config : diverses configuration système

```
[config]
ws type = 1080
day end hour = 21
gnuplot encoding = iso_8859_1
template encoding = iso-8859-1
language = en
logdata sync = 1
rain day threshold = 0.2
```

`ws type` est le "type" de station. Il devrait être fixé à `1080` pour la plupart des stations météo, ou `3080` si la console de votre station affiche l'éclairement solaire.

`day end hour` est la fin du "jour météorologique", selon l'heure locale sans tenir compte de l'heure d'été. Généralement, les valeurs sont 21, 9, ou 24.

`gnuplot encoding` est l'encodage de texte utilisé lors du tracé de graphes. La valeur par défaut est `iso_8859_1` permettant le symbole des degrés, fort utile dans une application météorologique ! D'autres valeurs peuvent être nécessaire si votre langue comporte des caractères accentués. Les valeurs possible dépendent de votre installation gnuplot, quelques expérimentations peuvent donc être nécessaire.

`template encoding` est l'encodage de text utilisé pour le gabarit. La valeur par défaut est `iso-8859-1`, ce qui est l'encodage utilisé dans les gabarits d'exemple. Si vous créez des gabarits avec un autre jeu de caractère, vous devez changer cette valeur pour celle de vos gabarits.

`language` est utilisé pour traduire pywws. Il est optionnel, puisque pywws utilise habituellement la langue par défaut de l'ordinateur tel que configuré par la variable d'environnement LANG. Vous trouverez les langages disponibles dans le sous-dossier `translations` de votre installation pywws. Si vous configurez une langue qui n'est pas inclue, pywws affichera en anglais.

`logdata sync` configure la qualité de la synchronisation utilisé par pywws.LogData. Fixer à 0 pour rapide & imprécis ou 1 pour plus lent, mais précis.

`rain day threshold` est la quantité de pluie (en mm) tombée en une journée pour qu'elle puisse être considéré comme un jour de pluie dans les données sommaires mensuels .

#### paths : dossiers dans lesquels sont emmagasinés les gabarits, etc.

```
[paths]
templates = /home/$USER/weather/templates/
graph_templates = /home/$USER/weather/graph_templates/
user_calib = /home/jim/weather/modules/usercalib
work = /tmp/weather
```

Ces trois entrées spécifient où sont emmagasinés vos gabarits de texte et de graphes, où doivent être créé les fichiers temporaires, et la localisation de votre module de calibration (si utilisé).

#### live : tâches à accomplir à chaque 48 secondes

```
[live]
services = ['underground']
twitter = []
text = []
plot = []
yowindow = /home/jim/data/yowindow.xml
```

Cette section spécifie les tâche devant être effectuées pour chaque échantillonnage pendant la 'journalisation temps réel', c.à.d. chaque 48 secondes. Il est peu probable que vous vouliez faire autre chose que le téléversement à Weather Underground ou la mise à jour du fichier YoWindow aussi souvent.

`services` est une liste de 'services' auxquels envoyer vos données. Chaque service listé doit avoir un fichier de configuration dans le dossier `pywws/services/`. Voir pywws.toservice pour plus de détails.

`twitter` est une liste de gabarit texte à traiter avant de les poster sur Twitter.

`text` et `plot` sont des listes de gabarits texte et graphe à traiter et à téléverser sur votre site web.

`yowindow` indique le chemin complet du fichier au format xml à générer pour le widget YoWindow weather (voir http://yowindow.com/). Si vous n'employez pas YoWindow, omettez cette entrée.

#### logged : tâches à accomplir chaque fois que la station enregistre une lecture

```
[logged]
services = ['underground', 'metoffice']
twitter = ['tweet.txt']
text = []
plot = []
```

Cette section spécifie les tâche devant être réalisées chaque fois qu'une donnée est lue en mode 'temps réel' ou chaque fois qu'une tâche horaire cron est exécutée.

`services` est une liste de 'services' auxquels envoyer vos données. Chaque service listé doit avoir un fichier de configuration dans le dossier `pywws/services/`. Voir pywws.toservice pour plus de détails.

`twitter` est une liste de gabarit texte à traiter avant de les poster sur Twitter.

`text` et `plot` sont des listes de gabarits texte et graphe à traiter et à téléverser sur votre site web.

#### hourly : tâches à réaliser à chaque heure

```
[hourly]
services = []
twitter = ['tweet.txt']
text = ['24hrs.txt', '6hrs.txt', '7days.txt', 'feed_hourly.xml', 'allmonths.txt']
plot = ['7days.png.xml', '24hrs.png.xml', 'rose_12hrs.png.xml']
```

Cette section spécifie les tâches à réaliser à chaque heure en mode 'temps réel' ou exécutées en tant que tâche horaire cron.

`services` est une liste de 'services' auxquels envoyer vos données. Chaque service listé doit avoir un fichier de configuration dans le dossier `pywws/services/`. Voir pywws.toservice pour plus de détails.

`twitter` est une liste de gabarit texte à traiter avant de les poster sur Twitter.

`text` et `plot` sont des listes de gabarits texte et graphe à traiter et à téléverser sur votre site web.

#### 12 hourly : tâches à réaliser à chaque 12 heures

```
[12 hourly]
services = []
twitter = []
text = []
plot = []
```

Cette section spécifie les tâches à réaliser à chaque 12 heures en mode 'temps réel' ou exécutées en tant que tâche horaire cron. Utilisez pour des choses qui ne changent pas très souvent, tel que les graphes mensuels.

`services` est une liste de 'services' auxquels envoyer vos données. Chaque service listé doit avoir un fichier de configuration dans le dossier `pywws/services/`. Voir pywws.toservice pour plus de détails.

`twitter` est une liste de gabarit texte à traiter avant de les poster sur Twitter.

`text` et `plot` sont des listes de gabarits texte et graphe à traiter et à téléverser sur votre site web.

#### daily : tâches à faire à chaque 24 heures

```
[daily]
services = []
twitter = []
text = ['feed_daily.xml']
plot = ['2008.png.xml', '2009.png.xml', '2010.png.xml', '28days.png.xml']
```

Cette section spécifie les tâches à réaliser à chaque jour en mode 'temps réel' exécutées en tant que tâche horaire cron. Utilisez pour des choses qui ne changent pas très souvent, tel que les graphes mensuels ou annuels.

`services` est une liste de 'services' auxquels envoyer vos données. Chaque service listé doit avoir un fichier de configuration dans le dossier `pywws/services/`. Voir pywws.toservice pour plus de détails.

`twitter` est une liste de gabarit texte à traiter avant de les poster sur Twitter.

`text` et `plot` sont des listes de gabarits texte et graphe à traiter et à téléverser sur votre site web.

#### ftp : configuration du téléversement à un site web

```
[ftp]
local site = False
secure = False
site = ftp.your_isp.co.uk
user = username
password = userpassword
directory = public_html/weather/data/
```

Ces entrées fournissent des détails sur votre site web (ou répertoire local) où doivent être transférés les fichier de texte et de graphe traités.

`local site` spécifie si les fichiers doivent être copiés dans un dossier local ou envoyé sur un site distant. Vous pouvez vouloir fixer ce paramètre si votre serveur web fonctionne sur le même ordinateur qui exécute pywws.

`secure` spécifie si le transfert des fichiers utilise SFTP (FTP sécurisé) au lieu de l'habituel FTP. Votre hébergeur doit être en mesure de vous dire si vous pouvez utiliser SFTP.

`site` est l'adresse web de votre site FTP à laquelle transférer vos fichiers.

`user` et `password` sont les informations de branchement pour votre site FTP. Votre hébergeur doit vous avoir fourni ces informations.

`directory` spécifie où sur le site FTP (ou le disque local) les fichiers doivent être envoyés. Notez que vous pourriez devoir expérimenter quelque peu - vous pourriez par exemple devoir utiliser un caractère '/' au début de votre adresse.

#### twitter : configuration pour poster sur Twitter

```
[twitter]
secret = longstringofrandomcharacters
key = evenlongerstringofrandomcharacters
latitude = 51.365
longitude = -0.251
```

`secret` et `key` sont les données d'authentification fournies par Twitter. Pour les configurer, exécutez le programme `TwitterAuth.py`.

`latitude` et `longitude` sont vos données de localisation et sont optionnelles. Si vous les incluez dans vos 'tweets météo' vos utilisateurs pourront voir où est située votre station. Ce qui pourrait aider les utilisateurs à trouver votre station s'ils cherchent par localisation.

#### underground, metoffice, temperaturnu etc : configuration pour poster sur ces 'services'

```
[underground]
station = IXYZABA5
password = secret
```

Ces sections contiennent l'information nécessaire pour envoyer vos données sur ces services météo, tel que mot de passe et ID de votre station. Le nom de ces entrées de données dépend du service. L'exemple montré affiche les informations pour Weather Underground.

`station` est le PWS ID (Identificateur de station) alloué à la station météo par Weather Underground.

`password` est votre mot de passe pour Weather Underground.

### status.ini - format du fichier de status

Ce fichier est crée par pywws et ne doit pas (habituellement) être édité. Les section suivantes sont actuellement utilisées
- fixed : valeurs copiées à partir des "blocs fixes" de la station météo.
- clock : information de synchronisation.
- last update : date et heure de la plus récente tâche complétée.

#### fixed : valeurs copiées à partir des "blocs fixes" de la station météo

```
[fixed]
pressure offset = 7.4
fixed block = {...}
```

`pressure offset` est la différence entre la pression de l'air absolue et relative ; copié à partir de la station météorologique, en supposant que vous l'avez configuré pour afficher la bonne pression relative.

`fixed block` est toutes les données stockées dans les 256 octets de mémoire de la station. Cela comprend les valeures minimales et maximum, les paramètres de seuil d'alarme, unités de mesure et ainsi de suite.

#### clock : information de synchronisation

```
[clock]
station = 1360322930.02
sensor = 1360322743.69
```

Ces valeurs enregistrent le temps calculé lorsque l'horloge de la station a enregistrée certaines données et lorsque les capteurs extérieurs ont transmis un nouveau jeu de données. Elles sont utilisées pour tenter d'empêcher le plantage de l'interface USB si l'ordinateur accède à la station météo en même temps que l'un ou l'autre de ces événements, un problème commun à de nombreuses stations compatibles EasyWeather. Les durées sont mesurées toutes les 24 heures pour permettre la dérive de l'horloge

#### last update : date et heure de la plus récente tâche complétée

```
[last update]
hourly = 2013-05-30 19:04:15
logged = 2013-05-30 19:04:15
daily = 2013-05-30 09:04:15
openweathermap = 2013-05-30 18:59:15
underground = 2013-05-30 18:58:34
metoffice = 2013-05-30 18:59:15
12 hourly = 2013-05-30 09:04:15
```

Ces enregistrement affichent la date et l'heure de la dernière exécution réussie des diverses tâches. Ils sont utilisés pour permettre aux tâches infructueuses (ex. panne de réseau empêchant les téléversements) d'être réessayé après quelques minutes.

### Indice Humidité (Humidex)

*Auteur de la section : Rodney Persky*

### Arrière-plan

L'utilisation de votre station météorologique peut être amusant, et les rapports quotidiens des diverses données des sites météo peuvent être très utiles pour vos voisins afin de vérifier les conditions climatiques. Cependant, à un moment donné, vous voudrez peut-être savoir quel sont les effets de la météo sur votre corps, et si il y a une façon de savoir quand il est bon ou non de travailler à l'extérieur.

Nous entrons ici dans un royaume entier de calculs basés sur l'énergie de transfert à travers les murs, et la résistance offerte par ceux-ci. Ce qui peut être une grande aventure de la connaissance, et peut vous faire économiser beaucoup d'argent, et découvrir comment l'énergie se déplace tout autour.

### Introduction

L'Humidex est un outil pour déterminer comment un corps humain réagit à la combinaison du vent, de l'humidité et de la température. Fondement de la différence de température entre votre corps et votre peau et est complémentaire à ISO 7243 "Hot Environments - Estimation of the heat stress on working man". Quelques remarques importantes,

- Ces indices sont fondés sur un certain nombre d'hypothèses qui peuvent résulter en une sur- ou sous-estimation de l'état de vos organes internes
- Une station météo personnelle peut ne pas afficher les conditions précises, et donc sur ou sous-estimer l'humidité, le vent ou la température.
- Le choix de vêtements a un effet sur la fatigue et la capacité des organes à rejeter la chaleur, un Indice d'humidité faible ne signifie pas que vous puissiez porter n'importe quoi
- Un entraînement personnel affectera la réaction du corps à une température changeante et l'expérience aidera à savoir quand arrêter de travailler
- La durée des activités qui peuvent être exécutées requiert la connaissance de l'intensité, ce qui ne peut pas être représenté par cet index

### Hypothèse

Un certain nombre d'hypothèses ont été faites pour effectuer ce travail qui va affecter directement sa facilité d'utilisation. Ces hypothèses n'ont pas été rendues disponible par Environnement Canada, qui sont les développeurs à l'origine de l'Humidex utilisé dans la fonction PYWWS cadhumidex. Il est suffisamment sûr cependant de dire que la suite aurait été certaines hypothèses

- Type de vêtement, épaisseur
- Surface de la peau exposée à l'air libre
- Exposition au soleil

Cependant, il y a un certain nombre d'hypothèses que pyws doit faire dans le calcul de l'Humidex :

- Les relevés d'humidité, de vent et de température sont exacts

Il y a aussi les hypothèses au sujet du type de corps des individus et de leur 'acclimatation'

- L'entraînement physique d'un individu affectera la réponse de son corps face à des changements de température
- L'expérience aidera à savoir quand arrêter de travailler

### Références importantes

Being Prepared for Summer - http://www.ec.gc.ca/meteo-weather/default.asp?lang=En&n=86C0425B-1

### Utilisation

La fonction est descriptivement nommé `cadhumidex` et a la température et l'humidité comme paramètres, essentiellement la fonction agit comme une conversion et peut être utilisée d'une manière simple :

```
<ycalc>cadhumidex(data['temp_out'],data['hum_out'])</ycalc>
```

En l'assemblant, j'ai ajouté des couleurs qui suivent les couleurs d'avertissement de base et les différentes échelles pour produire un graphique décent :

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<graph>
  <title>Humidity Index, Bands indicate apparent discomfort in standard on-site working conditions</t
  <size>1820, 1024</size>
  <duration>hours=48</duration>
  <xtics>2</xtics>
  <xformat>%H%M</xformat>
  <dateformat></dateformat>
  <plot>
    <yrange>29, 55</yrange>
    <y2range>29, 55</y2range>
    <ylabel></ylabel>
    <y2label>Humidex</y2label>
    <source>raw</source>
    <subplot>
      <title>Humidex</title>
      <ycalc>cadhumidex(data['temp_out'],data['hum_out'])</ycalc>
      <colour>4</colour>
      <axes>x1y2</axes>
    </subplot>
    <subplot>
      <title>HI > 54, Heat Stroke Probable</title>
      <ycalc>54</ycalc>
      <axes>x1y2</axes>
      <colour>1</colour>
    </subplot>
    <subplot>
      <title>HI > 45, Dangerous</title>
      <ycalc>45</ycalc>
      <axes>x1y2</axes>
      <colour>8</colour>
    </subplot>
    <subplot>
      <title>HI > 40, Intense</title>
      <ycalc>40</ycalc>
      <axes>x1y2</axes>
      <colour>6</colour>
    </subplot>
    <subplot>
      <title>HI > 35, Evident</title>
      <ycalc>35</ycalc>
      <axes>x1y2</axes>
      <colour>2</colour>
    </subplot>
    <subplot>
      <title>HI > 30, Noticeable</title>
      <ycalc>30</ycalc>
      <axes>x1y2</axes>
      <colour>3</colour>
    </subplot>
  </plot>
</graph>
```

#### N'exécutez pas la dernière mise à jour ?

Si vous n'utilisez pas ou ne désirez pas utiliser la dernière mise à jour, vous pouvez l'implémenter à l'aide d'un <ycalc> plus long, comme suit :

```
<ycalc>data['temp_out']+0.555*(6.112*10**(7.5*data['temp_out']/(237.7+data['temp_out']))*data['hum_ou
```

### 4.1.5 Programmes et des modules Python

Contenu :

| | |
|---|---|
| *pywws.Hourly* | Obtient les données météorologiques, les traite, prépare les fichiers de graphes & textes et le |
| *pywws.LiveLog* | Obtient les données météo, les stocke et les traite. |
| *pywws.Reprocess* | Régénère les données sommaires horaires et quotidiennes. |
| *pywws.TwitterAuth* | Authorise pywws to post to your Twitter account |
| *pywws.SetWeatherStation* | Régle certains paramètres de la station météo. |
| *pywws.TestWeatherStation* | Teste la connexion à la station météo. |
| *pywws.USBQualityTest* | Teste la qualité de la connexion USB avec votre station météo |
| *pywws.EWtoPy* | Convertir les données du format EasyWeather.dat vers le format de pywws. |
| *pywws.Tasks* | Routines pour effectuer des tâches courantes telles que tracer les gaphes ou téléverser des fic |
| *pywws.LogData* | Enregistrer l'historique de la station météo dans un fichier |
| *pywws.Process* | Générer les relevés horaire, quotidien et mensuels à partir des données brutes de la station m |
| *pywws.calib* | Calibre les données brutes de la station météo |
| *pywws.Plot* | Trace les graphes de données météorologiques selon une recette XML. |
| *pywws.WindRose* | Tracer une "rose des vents". |
| *pywws.Template* | Crée un fichier de données texte basé sur un gabarit. |
| *pywws.Forecast* | Prédit le temps futur en utilisant les données récentes. |
| *pywws.ZambrettiCore* | |
| *pywws.Upload* | Téléverser les fichiers sur un serveur web via ftp ou copie dans un répertoire local |
| *pywws.ToTwitter* | Poste un message sur Twitter |
| *pywws.toservice* | Poste une mise à jour météo pour des services tels que Weather Underground |
| *pywws.YoWindow* | Génére un fichier XML YoWindow. |
| *pywws.WeatherStation* | Obtient les données des stations météorologiques WH1080/WH3080 et compatibles. |
| *pywws.device_ctypes_hidapi* | Low level USB interface to weather station, using ctypes to access hidapi. |
| *pywws.device_cython_hidapi* | Interface USB de bas niveau de la station météo, utilisant cython-hidapi. |
| *pywws.device_pyusb1* | Interface USB de bas niveau de la station météo, utilisant PyUSB. |
| *pywws.device_pyusb* | Interface USB de bas niveau de la station météo, utilisant PyUSB. |
| *pywws.DataStore* | DataStore.py - enregistre les lectures dans des fichiers facilement accessibles |
| *pywws.TimeZone* | Fournit un couple d'objets `datetime.tzinfo` représentant l'heure locale et UTC. |
| *pywws.Localisation* | Localisation.py - fournit des traductions de phrases en langue locale |
| *pywws.calib* | Calibre les données brutes de la station météo |
| *pywws.conversions* | conversions.py est un ensemble de fonctions pour convertir les unités pywws natives |
| *pywws.Logger* | Code commun pour l'enregistrement d'informations et d'erreurs. |

#### pywws.Hourly

Obtient les données météorologiques, les traite, prépare les fichiers de graphes & textes et les téléverse sur un site web.

Exécuté à partir d'une tâche cron, en règle générale toutes les heures.

```
usage: python -m pywws.Hourly [options] data_dir
options are:
 -h or --help     display this help
 -v or --verbose  increase amount of reassuring messages
data_dir is the root directory of the weather data (e.g. $(HOME)/weather/data)
```

Ce script ne fait guère plus que d'appeler d'autres modules dans l'ordre pour obtenir les données de la station météo, les traiter, tracer des graphes, de générer des fichiers texte et transférer le résultat vers un site web.

Pour plus d'informations sur l'utilisation de `Hourly.py`, voir Comment configurer la journalisation horaire avec pywws.

#### Fonctions

| |
|---|
| *Hourly*(data_dir) |
| *main*([argv]) |

pywws.Hourly.**Hourly**(*data_dir*)

pywws.Hourly.**main**(*argv=None*)

### pywws.LiveLog

Obtient les données météo, les stocke et les traite.

Exécuter en continu, ayant définit quelles sont les tâches à accomplir.

```
usage: python -m pywws.LiveLog [options] data_dir
options are:
 -h      or --help      display this help
 -l file or --log file  write log information to file
 -v      or --verbose   increase amount of reassuring messages
data_dir is the root directory of the weather data (e.g. /data/weather)
```

Pour plus d'informations sur l'utilisation `LiveLog.py`, voir Comment configurer le mode 'live' avec pywws.

#### Fonctions

| |
|---|
| *LiveLog*(data_dir) |
| *main*([argv]) |

#### Classes

pywws.LiveLog.**LiveLog**(*data_dir*)

pywws.LiveLog.**main**(*argv=None*)

### pywws.Reprocess

Régénére les données sommaires horaires et quotidiennes.

```
usage: python -m pywws.Reprocess [options] data_dir
options are:
 -h | --help     display this help
 -v | --verbose  increase number of informative messages
data_dir is the root directory of the weather data
```

#### Introduction

Ce programme recrée les données récapitulatives calibrées, horaires, quotidiennes et mensuelles qui sont créées par le programme *pywws.Process*. Il doit être exécuté chaque fois que vous mettez à jour vers une nouvelle version de pywws (si le format des données sommaires a changé), changer votre module de calibration ou modifier votre décalage de pression.

#### API détaillé

#### Fonctions

| |
|---|
| *Reprocess*(data_dir) |
| *main*([argv]) |

pywws.Reprocess.**Reprocess**(*data_dir*)

pywws.Reprocess.**main**(*argv=None*)

### pywws.TwitterAuth

Authorise pywws to post to your Twitter account

```
usage: python -m pywws.TwitterAuth [options] data_dir
options are:
 -h or --help       display this help
data_dir is the root directory of the weather data
```

This program authorises *pywws.ToTwitter* to post to a Twitter account. You need to create an account before running `TwitterAuth`. It opens a web browser window (or gives you a URL to copy to your web browser) where you log in to your Twitter account. If the login is successful the browser will display a 7 digit number which you then copy to `TwitterAuth`.

See Comment configurer pywws pour poster des messages sur Twitter for more detail on using Twitter with pywws.

#### Fonctions

| |
|---|
| *TwitterAuth*(params) |
| *main*([argv]) |

pywws.TwitterAuth.**TwitterAuth**(*params*)

pywws.TwitterAuth.**main**(*argv=None*)

### pywws.SetWeatherStation

Régle certains paramètres de la station météo

```
usage: python -m pywws.SetWeatherStation [options]
options are:
 -h   | --help           display this help
 -c   | --clock          set weather station clock to computer time
                         (unlikely to work)
 -p f | --pressure f     set relative pressure to f hPa
 -r n | --read_period n  set logging interval to n minutes
 -v   | --verbose        increase error message verbosity
 -z   | --zero_memory    clear the weather station logged reading count
```

#### Fonctions

| |
|---|
| *bcd_encode*(value) |
| *main*([argv]) |

#### Classes

pywws.SetWeatherStation.**bcd_encode**(*value*)

pywws.SetWeatherStation.**main**(*argv=None*)

### pywws.TestWeatherStation

Teste la connexion à la station météo.

Ceci est un utilitaire simple pour tester la communication avec la station météo. Si cela ne fonctionne pas, alors il y a un problème qui doit être résolu avant d'essayer tout autre programme. Les problèmes possibles comprennent l'installation incorrecte des librairies USB. ou un problème de permissions. Le problème le plus improbable est que vous ayez oublié de connecter la station météo à votre ordinateur !

```
usage: python -m pywws.TestWeatherStation [options]
options are:
      --help           display this help
 -c | --change         display any changes in "fixed block" data
 -d | --decode         display meaningful values instead of raw data
 -h | --history count  display the last "count" readings
 -l | --live           display 'live' data
 -m | --logged         display 'logged' data
 -u | --unknown        display unknown fixed block values
 -v | --verbose        increase amount of reassuring messages
                       (repeat for even more messages e.g. -vvv)
```

#### Fonctions

| |
|---|
| *main*([argv]) |
| *raw_dump*(pos, data) |

pywws.TestWeatherStation.**raw_dump**(*pos*, *data*)

pywws.TestWeatherStation.**main**(*argv=None*)

### pywws.USBQualityTest

Teste la qualité de la connexion USB avec votre station météo

```
usage: python -m pywws.USBQualityTest [options]
options are:
 -h | --help           display this help
 -v | --verbose        increase amount of reassuring messages
                       (repeat for even more messages e.g. -vvv)
```

La liaison USB à ma station n'est pas fiable à 100%. Les données lues à partir de la station par l'ordinateur sont occasionnellement corrompues, peut-être par une interférence. J'ai essayé de résoudre ce problème en mettant une perle de ferrite autour du câble USB et le déplaçant des sources d'interférences possibles, tels que les disques durs externes. Le tout sans succès jusqu'à présent.

Ce programme teste la connexion USB en lisant de manière continue toute la mémoire de la station météo (sauf les parties qui peuvent être changeantes) à la recherche d'erreurs. Chaque bloc de 32 octets est lu deux fois, et si les deux lectures diffèrent, un message d'avertissement s'affiche. Sont également affichés le nombre de blocs lus, et le nombre d'erreurs rencontrées.

En général, j'obtiens une ou deux erreurs par heure, donc le test doit être exécuté pendant plusieurs heures pour produire une mesure utile. Notez que les autres programmes qui accèdent à la station météo (tel que pywws.Hourly ou pywws.LiveLog) ne doivent pas être exécutés alors que le test est en cours.

Si vous exécutez ce test et obtenez absolument aucune erreur, s'il vous plaît faites le moi savoir. Il y a quelque chose de spécial dans votre configuration et j'aimerais bien savoir ce que c'est

#### Fonctions

| |
|---|
| *main*([argv]) |

pywws.USBQualityTest.**main**(*argv=None*)

### pywws.EWtoPy

Converti les données du format EasyWeather.dat vers le format de pywws.

```
usage: python -m pywws.EWtoPy [options] EasyWeather_file data_dir
options are:
 -h or --help    display this help
EasyWeather_file is the input data file, e.g. EasyWeather.dat
data_dir is the root directory of the weather data
```

#### Introduction

Ce programme convertit les données du format utilisé par le programme EasyWeather fourni avec la station météo, vers le format utilisé par pywws. Il est utile si vous avez utilisé EasyWeather pendant un certain temps avant de découvrir pywws.

Le fichier `EasyWeather.dat` n'est utilisé que pour fournir les données précédent le début des données pywws. Étant donné que votre station météo dispose de sa propre mémoire, vous devez exécuter *pywws.LogData* avant *pywws.EWtoPy* pour minimiser l'utilisation du fichier EasyWeather.dat.

*pywws.EWtoPy* convertit les horodates EasyWeather.dat de l'heure locale vers l'heure UTC. Cela peut causer des problèmes lorsque l'heure d'été se termine, puisque l'heure locale semble revenir en arrière d'une heure. Le programme tente de détecter et de corriger les horodates concernées, mais je n'ai pas pu tester cela sur une variété de fuseaux horaires.

#### API détaillé

#### Fonctions

| |
|---|
| *main*([argv]) |

#### Classes

pywws.EWtoPy.**main**(*argv=None*)

### pywws.Tasks

Routines pour effectuer des tâches courantes telles que tracer les gaphes ou téléverser des fichiers.

#### Classes

| |
|---|
| *RegularTasks*(params, status, calib_data, ...) |

**class** pywws.Tasks.**RegularTasks**(*params*, *status*, *calib_data*, *hourly_data*, *daily_data*, *monthly_data*, *asynch=False*)

**stop_thread**()
**has_live_tasks**()
**do_live**(*data*)
**do_tasks**()
**do_twitter**(*template*, *data=None*)
**do_plot**(*template*)
**do_template**(*template*, *data=None*)

### pywws.LogData

Enregistre l'historique de la station météo dans un fichier

```
usage: python -m pywws.LogData [options] data_dir
options are:
 -h   | --help     display this help
 -c   | --clear    clear weather station's memory full indicator
 -s n | --sync n   set quality of synchronisation to weather station (0 or 1)
 -v   | --verbose  increase number of informative messages
data_dir is the root directory of the weather data
```

Ce programme / module reçoit les données de la mémoire de la station météo et les stockent dans un fichier. Chaque fois qu'il est lancé, il récupère toutes les données qui sont plus récentes que les dernières données enregistrées, de sorte qu'il ne doit être exécuté que toutes les heures. Comme la station météorologique stocke généralement deux semaines de lectures (en fonction de l'intervalle d'enregistrement), LogData.py pourrait être exécuté très rarement si vous n'avez pas besoin de données fréquemment mise à jour.

Il n'y a aucune information sur la date ou l'heure dans les données brutes de la station météo, alors LogData.py crée un horodatage pour chaque lecture. Il utilise l'horloge de l'ordinateur, plutôt que l'horloge de la station météo qui peut ne pas être lue correctement par l'ordinateur. L'horloge d'un Pc en réseau devrait être réglée avec précision par ntp.

La synchronisation avec la station météo est obtenue par l'attente d'un changement dans les données en cours. Il y a deux niveaux de synchronisation, fixés par le fichier de configuration ou une option de ligne de commande. Le niveau 0 est plus rapide, mais n'est exacte qu'à environ douze secondes. Le niveau 1 attend jusqu'à ce que la station météo enregistre un nouvel enregistrement, et obtient un horodatage précis sur quelque secondes. Notez que cela peut prendre un certain temps, si l'intervalle d'enregistrement est supérieur aux cinq minutes recommandées.

#### Fonctions

| | |
|---|---|
| *Catchup*(ws, logger, raw_data, last_date, ...) | |
| *CheckFixedBlock*(ws, params, status, logger) | |
| *LogData*(params, status, raw_data[, sync, clear]) | |
| *main*([argv]) | |

#### Classes

pywws.LogData.**Catchup**(*ws*, *logger*, *raw_data*, *last_date*, *last_ptr*)

pywws.LogData.**CheckFixedBlock**(*ws*, *params*, *status*, *logger*)

pywws.LogData.**LogData**(*params*, *status*, *raw_data*, *sync=None*, *clear=False*)

pywws.LogData.**main**(*argv=None*)

### pywws.Process

Génére des sommaires horaires, quotidiens et mensuels à partir des données brutes de la station météo

```
usage: python -m pywws.Process [options] data_dir
options are:
 -h or --help     display this help
```

```
 -v or --verbose  increase number of informative messages
data_dir is the root directory of the weather data
```

Ce module prend les données brutes de la station météo (typiquement échantillonnées tous les cinq ou dix minutes) et génère des sommaires horaires, quotidiens et mensuels, ce qui est utile pour créer des tableaux et des graphes.

Avant de calculer les résumés de données, les données brutes sont "calibrées" en utilisant une fonction programmable par l'utilisateur. Voir pywws.calib pour plus de détails.

Les données horaire sont dérivées de tous les enregistrements lus en une heure, par exemple 18 :00 :00-18 :59 :59, et tien compte de l'indice du dernier relevé terminée à cette heure.

Les données quotidiennes résument la météo sur une période de 24 heures se terminant généralement à 2100 ou 0900 heures, heure locale (non heure d'été), bien que minuit est une autre convention populaire. Il est également indexé par le dernier enregistrement complet de la période. Le jour et la nuit, tels qu'utilisés dans le calcul des températures maximales et minimales, sont supposées commencer à 0900 et 2100 heure locale, soit 1000 et 2200 lorsque l'heure d'été est en vigueur, quel que soit le jour météorologique.

Pour régler le jour météorologique à votre préférence, ou celle utilisée par votre station météorologique locale officielle, modifier la ligne "day end hour" dans votre fichier `weather.ini`, puis exécutez Reprocess.py pour régénérer les résumés.

Les données récapitulatives mensuelles sont calculées à partir des données sommaires quotidiennes. Si la journée météorologique ne s'arrête pas à minuit, alors chaque mois peut commencer et se terminer jusqu'à 12 heures avant ou après minuit.

La moyenne des données de vitesse du vent durant la dernière heure (ou jour) et la vitesse maximale des rafales pendant l'heure (ou jour) est enregistrée. La direction prédominante du vent est calculée en utilisant l'arithmétique vectorielle.

La pluie est convertie à partir des données brutes "totale depuis la dernière réinitialisation" chiffré à un total plus utile durant la dernière heure, un jour ou mois.

### Fonctions

| | |
|---|---|
| *Process*(params, status, raw_data, ...) | Génère les résumés de données brutes de la station météo. |
| *calibrate_data*(logger, params, status, ...) | Calibre les données brutes, en utilisant une fonction fournie par l'utilisateur |
| *generate_daily*(logger, day_end_hour, ...) | Génère les résumés quotidiens à partir des données calibrées et horaires. |
| *generate_hourly*(logger, calib_data, ...) | Génère les résumés horaire à partir des données calibrées. |
| *generate_monthly*(logger, rain_day_threshold, ...) | Génère les résumés mensuels à partir des données quotidiennes. |
| *main*([argv]) | |

### Classes

| | |
|---|---|
| *Average*() | Calcule la moyenne des multiples valeurs de données. |
| *DayAcc*(daytime) | 'Accumule' les données météorologiques pour produire le résumé quotidien. |
| *HourAcc*(last_rain) | 'Accumule' les données météorologiques brutes pour produire le récapitulatif horaire. |
| *Maximum*() | Calcule la valeur maximale ainsi que l'heure et la date des multiples valeurs de données. |
| *Minimum*() | Calcule la valeur minimale ainsi que l'heure et la date des multiples valeurs de données. |
| *MonthAcc*(rain_day_threshold) | 'Accumule' les données météorologiques quotidiennes pour produire le sommaire mensuel. |

**class** pywws.Process.**Average**

Calcule la moyenne des multiples valeurs de données.

**add**(*value*)

**result**()

**class** pywws.Process.**Minimum**

Calcule la valeur minimale ainsi que l'heure et la date des multiples valeurs de données.

**add**(*value*, *time*)

**result**()

**class** pywws.Process.**Maximum**

Calcule la valeur maximale ainsi que l'heure et la date des multiples valeurs de données.

**add**(*value*, *time*)

**result**()

**class** pywws.Process.**HourAcc**(*last_rain*)

'Accumule' les données météorologiques brutes pour produire le récapitulatif horaire.

Calcule la vitesse moyenne du vent et les rafales de vent maximales, trouve la direction des vents dominants et calcule les précipitations totales.

**reset**()

**add_raw**(*data*)

**result**()

**class** pywws.Process.**DayAcc**(*daytime*)

'Accumule' les données météorologiques pour produire le résumé quotidien.

Calcule la vitesse moyenne du vent, la rafale maximale et les températures maximum de jour et minimum de nuit, trouve la direction du vent dominant et calcule les précipitations totales.

Le jour est supposé être entre 0900-2100 et la nuit entre 2100-0900, heure locale (1000-2200 et 2200-1000 au cours des heures d'été), quelle que soit le réglage de "day end hour".

**reset**()

**add_raw**(*data*)

**add_hourly**(*data*)

**result**()

**class** pywws.Process.**MonthAcc**(*rain_day_threshold*)

'Accumule' les données météorologiques quotidiennes pour produire le sommaire mensuel.

Calcule la température maximum le jour et le minimum la nuit.

**reset**()

**add_daily**(*data*)

**result**()

pywws.Process.**calibrate_data**(*logger*, *params*, *status*, *raw_data*, *calib_data*)

Calibre les données brutes, en utilisant une fonction fournie par l'utilisateur.

pywws.Process.**generate_hourly**(*logger*, *calib_data*, *hourly_data*, *process_from*)

Génére les résumés horaire à partir des données calibrées.

pywws.Process.**generate_daily**(*logger*, *day_end_hour*, *daytime*, *calib_data*, *hourly_data*, *daily_data*, *process_from*)

Génére les résumés quotidiens à partir des données calibrées et horaires.

pywws.Process.**generate_monthly**(*logger*, *rain_day_threshold*, *day_end_hour*, *time_offset*, *daily_data*, *monthly_data*, *process_from*)

Génère les résumés mensuels à partir des données quotidiennes.

pywws.Process.**Process**(*params*, *status*, *raw_data*, *calib_data*, *hourly_data*, *daily_data*, *monthly_data*)

Génére les résumés de données brutes de la station météo.

La fin de la journée météorologique (généralement 2100 ou 0900 heure locale) se trouve dans le fichier de configuration `weather.ini`. La valeur par défaut est 2100 (2200 au cours de l'heure d'été), selon la convention historique pour les lectures de la station météo.

`pywws.Process.`**`main`**(*argv=None*)

### pywws.calib

#### Calibre les données brutes de la station météo

Ce module permet d'ajuster les données brutes de la station météo dans le cadre de l'étape de 'traitement' (voir pywws.Process ). Par exemple, si vous avez installé un entonnoir pour doubler votre zone de collecte du pluviomètre, vous pouvez écrire une routine de calibration pour doubler la valeur de pluie.

**La classe de calibration par défaut fait deux choses :**

1. Générer pression atmosphérique relative.
2. Retirer les valeurs invalides de direction du vent.

Toute calibration utilisateur que vous écrivez doit également faire ceci.

**Écrire votre module de calibration** Tout d'abord, décider où vous voulez garder votre module. Comme vos gabarits texte et graphe, il est préférable de le garder séparé du code pywws, de sorte qu'il n'est pas affecté par les mises à jour de pywws. Je suggère la création d'un répertoire `modules` au même endroit que votre répertoire `templates`.

Créez un fichier texte dans votre répertoire `modules`, par exemple, `Calib.py` et copiez-y le texte suivant :

```
class Calib(object):
    def __init__(self, status):
        self.pressure_offset = eval(status.get('fixed', 'pressure offset'))
    def calib(self, raw):
        result = dict(raw)
        # sanitise data
        if result['wind_dir'] is not None and result['wind_dir'] >= 16:
            result['wind_dir'] = None
        # calculate relative pressure
        result['rel_pressure'] = raw['abs_pressure'] + self.pressure_offset
        return result
```

La classe *`Calib`* a deux méthodes. `Calib.__init__()` est le constructeur et est un bon endroit pour mettre toutes les constantes nécessaires. :py : meth :*Calib.calib* génère un ensemble unique de données 'calibrées' à partir d'un seul ensemble de données 'brutes'. Il y a quelques règles à suivre lors de l'écriture de cette méthode :

- Assurez-vous d'inclure la ligne `result = dict(raw)`, qui permet de copier toutes les données brutes à votre résultat, au début.
- Ne modifiez pas les données brutes.
- Assurez-vous que vous définissez `result['rel_pressure']`.
- N'oubliez pas de retourner (`return`) le résultat à la fin.

Lorsque vous avez fini d'écrire votre module de calibration vous pouvez demander à pywws de l'utiliser en mettant son emplacement dans votre fichier `weather.ini`. Il va dans les sections `[paths]`, comme le montre l'exemple ci-dessous :

```
[paths]
work = /tmp/weather
templates = /home/jim/weather/templates/
graph_templates = /home/jim/weather/graph_templates/
user_calib = /home/jim/weather/modules/usercalib
```

Notez que la valeur de `user_calib` ne doit pas inclure le `.py` à la fin du nom de fichier.

### Classes

| | |
|---|---|
| *Calib*(params, status) | Classe qui implémente la calibration par défaut ou la calibration par l'utilisateur. |
| *DefaultCalib*(status) | Classe de calibration par défaut |

**class** pywws.calib.**DefaultCalib**(*status*)

Classe de calibration par défaut

Cette classe définit la pression relative, en utilisant un décalage de pression lues à partir de la station météorologique, et 'normalise' la valeur de la direction du vent. C'est la calibration strictement minimale nécessaire.

**calib**(*raw*)

**class** pywws.calib.**Calib**(*params*, *status*)

Classe qui implémente la calibration par défaut ou la calibration par l'utilisateur.

D'autres modules pywws utilisent cette méthode pour créer un objet de calibration. Le constructeur crée soit un objet de calibration par défaut ou un objet de calibration utilisateur, en fonction de la valeur `user_calib` dans la section `[paths]` du paramètre `params`. Il adopte alors la méthode de calibration de l'objet :py : meth :*calib* comme sien.

**calibrator = None**

## pywws.Plot

Trace les graphes de données météorologiques selon une recette XML.

```
usage: python -m pywws.Plot [options] data_dir temp_dir xml_file output_file
options are:
 -h or --help    display this help
data_dir is the root directory of the weather data
temp_dir is a workspace for temporary files e.g. /tmp
xml_file is the name of the source file that describes the plot
output_file is the name of the image file to be created e.g. 24hrs.png
```

### Introduction

Comme Template.py celui-ci est un des modules les plus difficiles à utiliser dans le logiciel de collecte de la station météo. Il trace un graphe (ou un ensemble de graphes) de données météorologiques. Presque tout le graphe est contrôlé par un fichier XML. Je me réfère à ces fichiers en tant que modèles, mais ils ne sont pas des gabarits dans le même sens que Template.py les utilise pour créer des fichiers texte.

Avant d'écrire vos propres fichiers de gabarit de graphe, il pourrait être utile d'examiner quelques exemples dans le répertoire example_graph_templates. Si (comme moi) vous n'êtes pas familier avec le langage XML, je vous suggère de lire le tutoriel XML du Site du Zéro.

**Syntaxe XML du fichier graphe** Voici le modèle de graphe utile le plus simple. Il trace la température extérieure pendant les 24 dernières heures.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<graph>
  <plot>
    <subplot>
      <title>Temperature (°C)</title>
```

```
      <ycalc>data['temp_out']</ycalc>
    </subplot>
  </plot>
</graph>
```

Dans cet exemple, l'élément racine du graphe a un élément de tracé, lequel a un élément graphe secondaire. L'élément graphe secondaire contient un élément de titre et un élément ycalc. Pour tracer davantage de données sur le même ensemble d'axes (par exemple le point de rosée et la température), on peut ajouter plus d'éléments sous-graphe. Pour tracer plus d'un ensemble d'axes (par exemple la vitesse du vent qui est mesurée en unités différentes de la température) dans le même fichier, nous pouvons ajouter plus d'éléments de graphe.

La hiérarchie complète de l'élément est illustré ci-dessous.

```
graph
    plot
        subplot
            xcalc
            ycalc
            axes
            style
            colour
            title
        bmargin
        yrange
        y2range
        ytics
        y2tics
        ylabel
        ylabelangle
        y2label
        y2labelangle
        grid
        source
        boxwidth
        title
        command
    start
    stop
    duration
    layout
    size
    fileformat
    terminal
    lmargin
    rmargin
    xformat
    xlabel
    dateformat
    xtics
    title
```

**graph** C'est l'élément racine du fichier XML de graphe. Il n'a pas à être appelé "graph ", mais il doit y avoir exactement un élément racine.

**plot** Chaque élément graphe doit contenir au moins un élément de tracé. Un graphe distinct est établi pour chaque élément de tracé, mais tous partagent le même axe X.

**start** Cet élément règle la date et l'heure de départ de l'axe X. Il est utilisé pour la construction d'un élément Python datetime. Par exemple, pour démarrer le graphe à midi (heure locale), le jour de Noel 2008 : `<start>year=2008, month=12, day=25, hour=12</start>`. La valeur par défaut est (stop - duration).

**stop** Cet élément règle la date et l'heure de fin de l'axe X. Il est utilisé pour la construction d'un élément Python datetime. Par exemple, pour terminer le graphe à 10h am (heure locale), le jour du Nouvel An : `<stop>year=2009, month=1, day=1, hour=10</stop>`. La valeur par défaut est (start + duration), à moins que start ne soit pas définit dans lequel cas l'horodate de la dernière lecture horaire de la station météo est employée.

**duration** Cet élément définit l'étendue de l'axe X du graphique, à moins que start (début) et stop (arrêt) soient tous deux définis. Il est utilisé dans le construction d'un objet Python timedelta. Par exemple, pour tracer une semaine : `<duration>weeks=1</duration>`. La valeur par défaut est hours=24.

**layout** Contrôle la disposition des tracés. Par défaut sur une seule colonne. L'élément de mise en page définit les lignes et les colonnes. Par exemple : `<layout>4, 2</layout>` utilisera une grille de 4 lignes et 2 colonnes.

**size** Définit les dimensions globales du fichier image contenant le graphe. Par défaut (dans une mise en page à simple colonne) une largeur de 600 pixels et une hauteur de 200 pixels pour chaque tracé est employé, donc la dimension d'un graphe avec quatre éléments tracés serait de 600 x 800 pixels. Tout élément de taille doit inclure à la fois la largeur et la hauteur. Par exemple : `<size>800, 600</size>` produira une image 800 pixels de large et 600 pixels de haut.

**fileformat** Définit le format du fichier d'image contenant le graphe. Par défaut png. Toute chaîne reconnue par votre installation de gnuplot peut convenir. Par exemple : `<fileformat>gif</fileformat>` va produire une image GIF.

**terminal** Permet un contrôle complet des paramètres 'terminal' de gnuplot. Vous pouvez utiliser cette option si vous tracez un format d'image inhabituel. Toute chaîne reconnue par la commande 'terminal' de votre installation gnuplot devrait faire. Par exemple : `<terminal>svg enhanced font "arial,9" size 600,800 dynamic rounded</ terminal>`. Ce paramètre spécifie à la fois la taille et le format de fichier.

**lmargin** Définit la marge gauche des graphes, c'est à dire la distance entre l'axe de gauche sur le bord gauche de la zone de l'image. Selon la documentation de gnuplot les unités de lmargin sont des largeurs de caractères. La valeur par défaut est de 5, ce qui devrait être OK dans la plupart des circonstances.

**rmargin** Définit la marge droite des graphes, c'est à dire la distance entre l'axe de droite à l'extrémité droite de la zone d'image. Selon la documentation de gnuplot les unités de rmargin sont des largeurs de caractères. La valeur par défaut est -1, ce qui spécifie le réglage automatique.

**xformat** Définit le format des étiquettes de date / heure xtic. La valeur est une chaîne de style format strftime. Par défaut elle dépend de la durée du graphe : 24 heures ou moins est "%H%M", 24 heures sur 7 jours est "%a% d" et 7 jours ou plus est "%Y/%m/%d".

**xlabel** Définit l'étiquette d'axe X. La valeur est une chaîne de style format strftime. Par défaut dépend de la durée du graphe : 24 heures ou moins est "Time (%Z)", de 24 heures à 7 jours est «Day» et 7 jours ou plus est «Date». Le datetime utilisé pour calculer ce départ, ce qui peut produire des résultats inattendus lorsqu'un graphique s'étend de début ou de fin DST.

**dateformat** Définit le format des étiquettes de date à chaque extrémité de l'axe X. La valeur est une chaîne de style format strftime. Par défaut est "%Y/%m/%d". L'étiquette de droite est uniquement affichée si elle diffère de la gauche. Pour avoir aucune étiquette, définissez un format vide : `<dateformat></dateformat>`

**xtics** Définit l'espacement des "tic" des marques sur l'axe X. La valeur est un nombre entier d'heures. La valeur par défaut est de laisser gnuplot définir l'intervalle approprié.

**title** Définit le titre du graphe. Une seule ligne de texte, par exemple : `<title>La météo du jour</ title>`. Ce titre apparaît tout en haut du graphe, en dehors de toute zone de traçage.

**subplot** Chaque élément du graphe doit contenir au moins un élément graphe secondaire. Une tracé distinct est établi pour chaque élément sous-graphe, mais tous partagent les même axes X et Y.

**bmargin** Définit la marge inférieure, c'est à dire la distance entre la partie inférieure de l'axe X et le bord du graphique (ou le graphe suivante). La valeur par défaut est de laisser gnuplot ajuster automatiquement, ce qui fonctionne bien la plupart du temps, mais vous pouvez ajuster la valeur en fonction de votre installation.

La valeur autorisée est un nombre non négatif réel. Sur mon installation 0.9 est une bonne valeur, définie comme suit : `<bmargin> 0.9</ bmargin>`.

**yrange** Définit les limites inférieure et supérieure de l'axe Y gauche. La valeur est tout ce qui est recconnu par gnuplot, généralement une paire de nombres. La valeur par défaut est de permettre à gnuplot de définir les valeurs appropriées, ce qui est peu susceptible d'être ce que vous voulez. Par exemple, pour tracer des températures typiques du Royaume-Uni sans valeur va hors du graphe : `<yrange>-10, 30</yrange>`. Notez que les virgules sont converties en deux points, donc `<yrange>10:30</ yrange>` serait équivalent.

Vous pouvez utiliser un astérisque pour laisser gnuplot choisir une valeur appropriée. Par exemple, pour automatiser l'échelle de valeur supérieure, tout en fixant la valeur inférieure à zéro, utilisez `<yrange>0:*</yrange>`.

**y2range** Définit les limites inférieure et supérieure de l'axe Y de droite. Le défaut est pour l'axe Y de droite d'être la même que celle de gauche, mais fixer une plage différente est utile en double axe de traçage.

**ytics** Contrôle les "tic" des marques sur l'axe Y de gauche. La valeur peut être tout ce qui est compris par gnuplot. Par exemple, pour définir l'espacement tic à 45 utiliser `<ytics>45</ ytics>`. Des choses plus complexes sont également possibles, par exemple étiqueter un diagramme de direction du vent avec des points de boussole, utilisez `<y2tics>('N' 0, 'E' 90, 'S' 180, 'W' 270, 'N' 360)</ y2tics>`.

**y2tics** Contrôle les "tic" des marques sur l'axe de droite. Le format est le même que celui de ytics. Le comportement par défaut consiste à copier les marques de graduation de gauche, mais sans étiquette.

**ylabel** Ajoute une étiquette à gauche de l'axe Y. Par exemple, lors du tracé de la température : `<ylabel>°C</ylabel>`. Si vous utilisez ylabel vous voudrez probablement ajuster lmargin.

**ylabelangle** Ajuste l'angle de l'étiquette à gauche de l'axe Y, si votre version de gnuplot le soutient. Par exemple, pour écrire l'étiquette horizontalement : `<ylabelangle>90</ylabelangle>`.

**y2label** Ajoute une étiquette à l'axe Y de droite. Par exemple, lors du tracé de l'humidité : `<y2label>%</y2label>`. Cela est principalement utilisé lors du traçage de graphes à deux axes. Si vous utilisez y2label vous voudrez probablement ajuster rmargin.

**y2labelangle** Ajuste l'angle de l'étiquette d'axe Y à droite, si votre version de gnuplot le permet. Par exemple, pour écrire l'étiquette horizontalement : `<y2labelangle>90</ y2labelangle>`.

**grid** Ajoute une grille au tracé. Dans la plupart des situations, la grille par défaut de gnuplot est appropriée, si aucune valeur n'est nécessaire : `<grid></grid>`. Plus de contrôle est possible en utilisant l'une des options comprises par la commande set grid de gnuplot. Par exemple, pour avoir les lignes horizontales de la grille uniquement : `<grid>ytics</grid>`.

**source** Sélectionne les données météorologiques à tracer. Les valeurs autorisées sont `<source>raw</source>`, `<source> hourly</ source>`, `<source>daily</source>` et `<source>monthly</source>`. LA valeur par défaut est raw. Notez que les sources différentes ont des dictionnaires de données différents, donc ce choix affecte ycalc.

**boxwidth** Définit la largeur des "boîtes" utilisés lors de l'élaboration des graphes à barres. La valeur est une expression entière pour obtenir un nombre de secondes. Le défaut dépend de la source : brut est de 240, horaire est de 2800 et quotidien est de 2800 * 24.

**title** Définit le titre du graphe. Une seule ligne de texte, par exemple : `<title>Température (°C)</ title>`. Ce titre apparaît dans la zone de traçage, au-dessus de tous les titres sous-graphes.

**command** Exécute n'importe quelle commande gnuplot, juste avant la principale commande de tracé. Cette option permet aux utilisateurs avancés d'avoir plus de contrôle sur l'apparence du graphe. La valeur est une commande gnuplot valide, généralement commençant par un ensemble de mots. Par exemple : `<command>set key tmargin center horizontal width 1 noreverse enhanced autotitles box linetype −1 linewidth 1</ command>`. (Ne me demandez pas ce que cet exemple produit - Je ne suis pas un utilisateur avancé).

**xcalc** Contrôle le positionnement de l'axe X du tracé des valeurs de données. La valeur par défaut de data['idx'] est correcte pour la plupart des données, mais il y a quelques exceptions. Par exemple, lors du tracé des diagrammes à barres de pluies horaires, il est bon de centrer les barres sur 30 minutes après l'heure : `<xcalc>data ['idx'].replace(minute=30, second=0)</ xcalc>`.

**ycalc** Sélectionne les données à tracer. Toute expression Python sur une ligne retournant une valeur flottante simple peut être utilisée. Au plus simple ceci sélectionne une seule valeur du dictionnaire "data", par exemple : `<ycalc>data['temp_out']</ycalc>` trace la température extérieure. Des expressions plus complexes sont possibles, et certaines fonctions d'aide sont fournies. Par exemple : `<ycalc>dew_point(data['temp_out'], data['hum_out'])</ ycalc>` trace le point de rosée, et `<ycalc>wind_mph(data['wind_ave'])</ycalc>` trace la vitesse moyenne du vent en miles par heure.

Les courbes cumulatives sont également possibles. Le résultat de chaque calcul ycalc est stocké et mis à la disposition du calcul suivant dans la variable last_ycalc. Ceci peut être utilisé avec toute donnée, mais il est plus utile avec des précipitations : `<ycalc>data ['rain'] + last_ycalc</ ycalc>`.

**axes** Sélectionne contre quel axe Y les données sont tracées. Par défaut c'est l'axe de gauche, mais l'axe de droite peut être choisie avec : `<axes>x1y2</ axes>`. Ceci peut être utilisé en conjonction avec y2range pour tracer deux quantités indépendantes sur une courbe, par exemple la température et de l'humidité.

**style** Définit le style de ligne pour le graphe. Le défaut est une ligne continue lisse, d'épaisseur 1. Pour sélectionner un graphique à barres, utilisez : `<style>box</ style>`. Pour sélectionner des points sans ligne de raccordement, utilisez : `<style>+</ style>` ou `<style>x</ style>`. Pour sélectionner une épaisseur de ligne à 3 (par exemple) : `<style>line 3</ style>`. L'épaisseur des points peut être réglé de la même façon. Pour un contrôle complet (pour utilisateurs avancés) un style gnuplot complet peut être réglé ainsi : `<style>smooth unique lc 5 lw 3</ style>`.

**colour** Définit la couleur de la ligne de sous-graphe ou des boîtes. Toute valeur entière est acceptée. La charte des couleurs est fixée par gnuplot. La valeur par défaut est la couleur précédente plus un.

**title** Définit le titre du sous-graphe. Une seule ligne de texte, par exemple : `<title>Température (°C)</ title>`. Ce titre apparaît dans la zone de graphe, à côté d'un court segment de la couleur de la ligne utilisée pour le sous-graphe.

### API détaillé

### Fonctions

| | |
|---|---|
| *main*([argv]) | |

### Classes

| | |
|---|---|
| *BasePlotter*(params, status, raw_data, ...) | |
| *GraphPlotter*(params, status, raw_data, ...) | |
| *Record* | |

**class** pywws.Plot.**BasePlotter**(*params*, *status*, *raw_data*, *hourly_data*, *daily_data*, *monthly_data*, *work_dir*)

**DoPlot**(*input_file*, *output_file*)

**GetChildren**(*node*, *name*)

**GetValue**(*node*, *name*, *default*)

**class** pywws.Plot.**Record**

**class** pywws.Plot.**GraphPlotter**(*params*, *status*, *raw_data*, *hourly_data*, *daily_data*, *monthly_data*, *work_dir*)

**GetPlotList**()

**GetDefaultRows**()

**GetDefaultPlotSize**()

**GetPreamble**()

**PlotData**(*plot_no*, *plot*, *source*)

pywws.Plot.**main**(*argv=None*)

### pywws.WindRose

Tracer une "rose des vents".

```
usage: python -m pywws.WindRose [options] data_dir temp_dir xml_file output_file
options are:
 -h or --help    display this help
data_dir is the root directory of the weather data
temp_dir is a workspace for temporary files e.g. /tmp
xml_file is the name of the source file that describes the plot
output_file is the name of the image file to be created e.g. 24hrs.png
```

#### Introduction

Cette routine trace une ou plusieurs "rose des vents" (voir *Wikipedia <http ://en.wikipedia.org/wiki/Wind_rose> _* pour une description). Comme *pywws.Plot* presque tout est contrôlé par un fichier "recette"/gabarit XML.

Avant d'écrire vos propres fichiers gabarits, il pourrait être utile de jeter un coup d'oeil sur quelques exemples dans le répertoire example_graph_templates. Si (comme moi) vous n'êtes pas familier avec le langage XML, je vous suggère de lire *Le point sur le XML < http ://www.siteduzero.com/tutoriel-3-33440-le-point-sur-xml.html/> _*.

#### Syntaxe XML du fichier graphe

Voici le plus simple gabarit de rose des vents utile. Il trace le vent au cours des dernières 24 heures.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<graph>
  <windrose>
    <ycalc>data['wind_ave']</ycalc>
  </windrose>
</graph>
```

Dans cet exemple, l'élément racine du graphe est un élément de rose des vents qui ne contient rien de plus qu'un élément ycalc.

La hiérarchie complète de l'élément est illustré ci-dessous.

```
graph
    windrose
        xcalc
        ycalc
        threshold
        colour
        yrange
        points
        source
        title
    start
    stop
    duration
    layout
    size
    fileformat
    lmargin, rmargin, tmargin, bmargin
    title
```

**graph** C'est l'élément racine du fichier XML de graphe. Il n'a pas à être appelé "graph ", mais il doit y avoir exactement un élément racine.

**windrose** Un tracé séparé est établi pour chaque élément de rose des vents, mais tous partagent le même période.

**start** Cet élément règle la date et l'heure des roses des vents. Il est utilisé dans la construction d'un objet datetime Python. Par exemple, pour commencer, à midi (heure locale) le jour de Noël 2008 : `<start>year=2008, month=12, day=25, hour=12</ start>`. La valeur par défaut est (stop - durée).

**stop** Cet élément règle la date et l'heure de la fin des roses des vents. Il est utilisé dans la construction d'un objet datetime Python. Par exemple, pour se terminer à 10 h (heure locale) le jour de l'an 2009 : `<stop>year=2009, month=1, day=1, hour=10</stop>`. La valeur par défaut est (départ + durée), à moins que départ ne soit pas défini. Dans ce cas, l'horodatage de la dernière lecture horaire de la station est utilisée.

**duration** Cet élément définit la durée de la rose des vents, à moins que démarrage et arrêt soient définies. Il est utilisé dans la construction d'un objet Python timedelta. Par exemple, pour tracer une semaine : `<duration> weeks=1</duration>`. La valeur par défaut est hours=24.

**layout** Contrôle la disposition des graphes. Le défaut est une grille qui est plus large que haute. L'élément de mise en page définit les lignes et les colonnes. Par exemple : `<layout> 4, 2 </ layout>` utilisera une grille de 4 lignes et 2 colonnes.

**size** Définit les dimensions globales du fichier image contenant le graphe. Par défaut elle est d'une hauteur de 600 pixels et une largeur qui dépend de la disposition. Tout élément de taille doit inclure à la fois la largeur et la hauteur. Par exemple : `<size>800, 600</size>` produira une image 800 pixels de large et 600 pixels de haut.

**fileformat** Définit le format d'image du fichier contenant les graphes. Par défaut png. Toute chaîne reconnue par votre installation de gnuplot peut convenir. Par exemple : `<fileformat>gif</fileformat>` va produire une image GIF.

**lmargin, rmargin, tmargin, bmargin** Remplace automatiquement la marge calculée pour la gauche, la droite, la marge supérieure ou inférieure. Fournir tout nombre réel positif, par exemple "<lmargin>1.3</lmargin> ". Quelques essais peuvent être nécessaires pour trouver les meilleures valeurs.

**title** Définit le titre principal des graphes. Une seule ligne de texte, par exemple : `<title>La météo du jour</title>`. Ce titre apparaît tout en haut, en dehors de toute zone de graphe.

**xcalc** Sélectionne si les données sont incluses dans la rose des vents. La valeur doit être une expression logique Python valide. Par exemple, pour tracer une rose des vents pour l'après-midi seulement : `<xcalc>data ['idx'].hour&gt;= 12</xcalc>`. Cela permet l'agrégation des données de vent l'après-midi pendant plusieurs jours. Rappelez-vous que les données sont indexées en UTC, vous devez donc utiliser une expression qui tient compte de votre fuseau horaire. La valeur par défaut est "True".

**ycalc** Sélectionne les données à tracer. Toute expression Python sur une ligne qui retourne une valeur flottante simple peut être utilisée. Dans sa plus simple utilisation cela sélectionne une seule valeur du dictionnaire de "données", par exemple : `<ycalc>data['wind_ave']</ycalc>`. Pour convertir en mph, utiliser : `<ycalc>data['wind_ave'] * 3.6 / 1.609344</ycalc>`. Il est très peu probable de vouloir utiliser pour autre chose que 'wind_ave' ici.

**threshold** Définit les seuils pour chaque couleur sur les pétales de rose. Par défaut sont basées sur l'exemple Wikipedia. Les valeurs doivent être une liste correctement ordonnée de nombres réels, par exemple : `<threshold>0.5, 3.5, 7.5, 12.5, 18.5, 24.5, 31.5</threshold>` se rapproche de l'échelle de Beaufort, si ycalc a été défini pour convertir les vitesses de vent en mph.

**colour** Définit les couleurs des segments de seuil des pétales. Toute séquence de valeurs entières est acceptée. La correspondance de couleurs-nombres est définie par gnuplot. Les valeurs par défaut sont 0, 1, 2, 3, etc

**yrange** Définit les limites supérieures des axes. La rose indique le pourcentage du temps où le vent est venu d'une direction particulière. Par exemple, si vous vivez dans un endroit avec un vent très régulier vous voudrez peut-être permettre à des pourcentages plus élevés que la normale : `<yrange> 91</ yrange>` . Auto-mise à l'échelle est également possible, à l'aide d'un astérisque :`<yrange> *</ yrange>`

**points** Définit le texte des points cardinaux. Les valeurs par défaut sont 'N', 'S', 'E' et 'W'. Pour les graphiques dans une autre langue, vous pouvez les remplacer, par exemple par ceci : `<points>'No', 'Zu', 'Oo', 'We'</points>`. (La meilleure façon de le faire est de créer un fichier de langue, voir Localisation.py.)

**source** Sélectionne les données météorologiques à tracer. Les valeurs autorisées sont `<source>raw</source>`, `<source> hourly</ source>`, `<source>daily</source>` et `<source>monthly</source>`. LA valeur par défaut est raw. Notez que les sources différentes ont des dictionnaires de données différents, donc ce choix affecte ycalc.

**title** Définit le titre du graphe. Une seule ligne de texte, par exemple : `<title>Vents du matin</title>`. Ce titre apparaît dans la zone de traçage, au-dessus de la légende des couleurs.

### API détaillé

### Fonctions

| |
|---|
| *main*([argv]) |

### Classes

| |
|---|
| *RosePlotter*(params, status, raw_data, ...) |

**class** pywws.WindRose.**RosePlotter**(*params*, *status*, *raw_data*, *hourly_data*, *daily_data*, *monthly_data*, *work_dir*)

**GetPlotList**()

**GetDefaultRows**()

**GetDefaultPlotSize**()

**GetPreamble**()

**PlotData**(*plot_no*, *plot*, *source*)

pywws.WindRose.**main**(*argv=None*)

### pywws.Template

Crée un fichier de données texte basé sur un gabarit

```
usage: python -m pywws.Template [options] data_dir template_file output_file
options are:
 --help    display this help
data_dir is the root directory of the weather data
template_file is the template text source file
output_file is the name of the text file to be created
```

#### Introduction

C'est probablement le module le plus difficile à utiliser dans le logiciel de collecte de la station météo. Il génère des fichiers de texte basé sur un fichier "modèle" ainsi que les données brutes, horaires, quotidiennes et mensuelles de la station météo. Le traitement de gabarits va au-delà de la simple substitution des valeurs pour inclure des boucles, des sauts vers l'avant ou vers l'arrière dans les données, le traitement des données et de substitution des valeurs manquantes.

Un fichier gabarit peut être n'importe quel type de fichier texte (texte brut, xml, html, etc) dans lequel des "instructions de traitement" ont été ajoutées. Ces instructions de traitement sont délimitées par le caractère dièse ('#'). Ils ne sont pas affichés, mais provoquent autre chose : soit une valeur de données est insérée ou un nombre limité d'actions sont exécutées.

Avant d'écrire vos propres fichiers gabarits, il pourrait être utile d'examiner quelques-uns des exemples dans le répertoire example_templates.

#### Instructions de traitement

**##** affiche un seul caractère '#'.

**#! comment text#** un commentaire, pas de sortie généré. '' `comment text` peut être n'importe quel texte sans saut de ligne.

**#monthly#** affiche le sommaire des données "mensuelles". L'indice est réinintialisé à la plus récente valeur

**#daily#** affiche le sommaire des données "quotidiennes". L'indice est réinintialisé à la plus récente valeur

**#hourly#** affiche le sommaire des données "horaires". L'indice est réinintialisé à la plus récente valeur.

**#raw#** affiche les données "brutes". L'indice est réinitialisé à la plus récente valeur.

**#timezone name#** converti toute les valeurs de temps en zone de temps `name` avant affichage. Les valeurs permises sont `utc` ou `local`.

**#roundtime expr#** met l'arrondissement du temps en fonction ou hors fonction, selon `expr`. Lorsque l'arrondissement du temps est en fonction, 30 secondes est ajouté à chaque valeur de temps utilisée. Ceci est utile lorsque vous affichez seulement en heures et minutes, exemple avec un format "%H :%M", et que vous souhaitez qu'une valeur comme 10 :23 :58 apparaisse ainsi "10 :24". Pour la valeur de expr, utiliser `"True"` ou `"False"`.

**#jump count#** saute `count` valeurs. L'indice des données est ajusté par `count` heures ou jours. Des valeures négatives saute en arrière dans le temps.

Dans une boucle, c'est une bonne idée de mettre des sauts à la fin, juste avant l'instruction `#endloop`. La boucle peut alors se terminer proprement si elle a épuisé toutes les données.

**#goto date-time#** aller à `date-time`. L'indice des données est ajusté à l'enregistrement immédiatement après `date-time`. Ceci peut être en temps UTC ou dans votre zone de temps local, selon la configuration de `timezone`, et doit exactement correspondre au format de date ISO, par exemple `"2010-11-01 12:00:00"` est midi le 1er Novembre 2010.

Des parties de `date-time` peuvent être remplacées par le caractère de formatage % comme dans strftime pour spécifier l'indice courant de la bouche. Par exemple, `"%Y-%m-01 12:00:00"` est midi le 1er de ce mois.

**#loop count#** démarre une boucle qui se répétera `count` fois. `count` doit être égale à 1 ou plus.

**#endloop#** termine une boucle démarrée par `#loop count# ``. Le traitement du gabarit reviendra à la ligne contenant l'instruction ``#loop count#`. Ne pas essayer d'imbriquer des boucles.

**#key fmt_string no_value_string conversion#** affiche la valeure d'une donnée. `key` est la clé de données, par exemple `temp_out` pour la température extérieure. `fmt_string` est une chaîne de formatage comme printf (actuellement, l'opérateur % de Python), sauf pour les valeurs datetime, quand il est entré dans la méthode datetime `strftime() ``. ``no_value_string` est affiché au lieu de `fmt_string` lorsque la valeur de la donnée est absente, par exemple si la station a perdu contact avec la sonde de température extérieure. `conversion ` est une expression Python pour convertir les données, par exemple, pour convertir la vitesse du vent de m/s en mph, vous pouvez utiliser ``"x * 3.6 / 1.609344"`, ou la fonction fournie plus pratique `"wind_mph(x)"`.

Toutes ces valeurs doivent entre guillemets "si elles contiennent des espaces ou d'autres caractères potentiellement difficiles. Tout sauf `key` sont facultatifs, mais notez que si vous voulez spécifier une conversion, vous devez également spécifier `fmt_string` et `no_value_string`.

**#calc expression fmt_string no_value_string conversion#** affiche une valeur calculée à partir d'un ou plusieurs éléments de données. `expression` représente toute expression Python valide, par exemple, `"Dew_point(data['temp_out'], data['hum_out'])"` pour calculer le point de rosée. `fmt_string`, `no_value_string` et `conversion` sont tel que décrit ci-dessus. Notez qu'il est probablement plus efficace d'intégrer toute conversion dans l'expression.

#### Exemple

Voici un extrait qui montre un exemple de l'utilisation de base et avancé des caractéristiques du gabarit. Il fait partie du fichier modèle 6hrs.txt, ce qui génère un tableau HTML de 7 relevés horaires (qui devrait s'étendre sur 6 heures).

```
#hourly#
#jump -6#
#loop 7#
  <tr>
    <td>#idx "%Y/%m/%d" "" "[None, x][x.hour == 0 or loop_count == 7]"#</td>
    <td>#idx "%H%M %Z"#</td>
    <td>#temp_out "%.1f °C"#</td>
    <td>#hum_out "%d%%"#</td>
    <td>#wind_dir "%s" "-" "winddir_text(x)"#</td>
    <td>#wind_ave "%.0f mph" "" "wind_mph(x)"#</td>
    <td>#wind_gust "%.0f mph" "" "wind_mph(x)"#</td>
    <td>#rain "%0.1f mm"#</td>
    <td>#rel_pressure "%.0f hPa"#, #pressure_trend "%s" "" "pressure_trend_text(x)"#</td>
  </tr>
#jump 1#
#endloop#
```

Les trois premières lignes de cet extrait, exécutent ce qui suit : sélectionne les données horaires, revient en arrière de 6 heures, commence une boucle avec un nombre de 7. Un saut d'une heure apparaît juste avant la fin du segment répété. Comme ce dernier saut (d'une heure) se produit à chaque tour de la boucle, une séquence de 7 lectures de données sera affichée. La dernière ligne marque la fin de la boucle - tout ce qui est entre les lignes `#loop 7#` et `#endloop#` est affiché 7 fois.

Les instructions `#temp_out ...#`, `#hum_out ...#`, `#rain ... #` et `#rel_pressure ...#` affichent les données de base. Elles utilisent chacune un format de donnée `fmt_string` pour formater les données de façon appropriée. Les instructions''#wind_ave ...#'' et `#wind_gust ...#` montrent comment utiliser une expression de conversion pour convertir m/s en mph.

Les instructions `#wind_dir ... #` et `#pressure_trend ... #` affichent l'utilisation des fonctions `winddir_text` et `pressure_trend_text` pour convertir des valeurs numériques en texte .

Enfin nous arrivons aux valeurs de datetime. L'instruction `#idx "%H%M"#` affiche simplement l'heure (au format HHMM) de l'indice de donnée. L'instruction `#idx "%Y/%m/%d" "" "[None, x] [x.hour == 0 or loop_count == 7]"#` est un peu plus compliquée. Elle affiche la date, mais seulement sur la première ligne ou si la date a changée. Elle le fait en indexant le tableau `[None, x]` avec une expression booléenne vraie quand `loop_count` est égal à 7 (c'est à dire sur le premier passage dans la boucle) ou `x.hour` est zéro (c'est la première heure de la journée).

#### API détaillé

#### Fonctions

*main*([argv])

#### Classes

*Template*(params, status, calib_data, ...[, ...])

**class** pywws.Template.**Template**(*params*, *status*, *calib_data*, *hourly_data*, *daily_data*, *monthly_data*, *use_locale=True*)

**process**(*live_data*, *template_file*)

**make_text**(*template_file*, *live_data=None*)

**make_file**(*template_file*, *output_file*, *live_data=None*)

pywws.Template.**main**(*argv=None*)

### pywws.Forecast

Prédit le temps futur en utilisant les données récentes.

```
usage: python -m pywws.Forecast [options] data_dir
options are:
 -h | --help  display this help
data_dir is the root directory of the weather data
```

#### Fonctions

| | |
|---|---|
| *Zambretti*(params, hourly_data) | |
| *ZambrettiCode*(params, hourly_data) | |
| *main*([argv]) | |

#### Classes

pywws.Forecast.**ZambrettiCode**(*params*, *hourly_data*)

pywws.Forecast.**Zambretti**(*params*, *hourly_data*)

pywws.Forecast.**main**(*argv=None*)

### pywws.ZambrettiCore

#### Fonctions

| | |
|---|---|
| *ZambrettiCode*(pression, mois, vent, tendance) | Implémentation simple de l'algorithme de prévision de Zambretti. |
| *ZambrettiText*(lettre) | |
| *main*([argv]) | |

pywws.ZambrettiCore.**ZambrettiCode**(*pressure*, *month*, *wind*, *trend*, *north=True*, *baro_top=1050.0*, *baro_bottom=950.0*)

Mise en œuvre simple de l'algorithme de prévision Zambretti. Inspiré par algorithme Java de betel-juice.com, telle que convertie en Python par honeysucklecottage.me.uk, et d'autres informations à partir de http://www.meteormetrics.com/zambretti.htm

pywws.ZambrettiCore.**ZambrettiText**(*letter*)

pywws.ZambrettiCore.**main**(*argv=None*)

### pywws.Upload

Téléverse des fichiers sur un serveur web via ftp ou les copies dans un répertoire local

```
usage: python -m pywws.Upload [options] data_dir file [file...]
options are:
 -h or --help    display this help
data_dir is the root directory of the weather data
file is a file to be uploaded
```

Le nom d'usager et les détails du site ftp sont lues à partir du fichier weather.ini dans data_dir.

### Introduction

Ce module téléverse des fichiers sur (en général) un site Web *via* ftp/sftp ou copie les fichiers dans un répertoire local (par exemple, si vous exécutez pywws sur votre propre serveur web). Les détails de destination de téléversement sont stockés dans le fichier `weather.ini` dans votre répertoire de données. La seule façon de régler ces détails est d'éditer le fichier. Exécuter *pywws.Upload* une fois pour définir les valeurs par défaut, que vous pouvez ensuite modifier. Voici ce que vous êtes susceptible de trouver lorsque vous éditez `weather.ini` :

```
[ftp]
secure = False
directory = public_html/weather/data/
local site = False
password = secret
site = ftp.username.your_isp.co.uk
user = username
```

Ce sont, je l'espère, des options assez évidentes. `local site` vous permet de passer de téléversement sur un site distant, à la copie vers un site local. Si vous réglez `local site = True`, vous pouvez supprimer les lignes‘‘secure‘‘, `site`, `user` et `password`.

`directory` est le nom d'un répertoire dans lequel tous les fichiers téléversés seront déposés. Ce qui dépendra de la structure de votre site et du type d'hôte que vous utilisez. Votre fournisseur d'hébergement doit être en mesure de vous dire ce que vous devez utiliser pour `site` et `user`. Vous devriez déjà avoir choisi un `password`.

L'option `secure` vous permet de passer de ftp normal à sftp (ftp sur ssh). Certains fournisseurs d'hébergement l'offre comme mécanisme de chargement plus sûr, alors vous devriez probablement l'utiliser lorsque disponible.

### API détaillé

### Fonctions

*main*([argv])

### Classes

*Upload*(params)

**class** pywws.Upload.**Upload**(*params*)

**connect**()
**upload_file**(*file*)
**disconnect**()
**upload**(*files*)

pywws.Upload.**main**(*argv=None*)

### pywws.ToTwitter

Poster un message sur Twitter

```
usage: python -m pywws.ToTwitter [options] data_dir file
options are:
 -h | --help  display this help
data_dir is the root directory of the weather data
file is the text file to be uploaded
```

Ce module poste un bref message sur *Twitter <https ://twitter.com/> _*. Avant de poster sur Twitter, vous devez créer un compte et autoriser pywws en exécutant le programme :py : mod : *TwitterAuth*. Voir :doc : *../Guides/twitter* obtenir les instructions détaillées.

#### Fonctions

| |
|---|
| *main*([argv]) |

#### Classes

| |
|---|
| *ToTwitter*(params) |

**class** pywws.ToTwitter.**ToTwitter**(*params*)

**Upload**(*tweet*)
**UploadFile**(*file*)

pywws.ToTwitter.**main**(*argv=None*)

### pywws.toservice

Poster des mise à jour météo pour des services tels que Weather Underground

```
usage: python -m pywws.toservice [options] data_dir service_name
options are:
 -h or --help     display this help
 -c or --catchup  upload all data since last upload
 -v or --verbose  increase amount of reassuring messages
data_dir is the root directory of the weather data
service_name is the service to upload to, e.g. underground
```

#### Introduction

Plusieurs organisations permettent aux stations météorologiques de télécharger des données en utilisant une simple requête 'POST' ou 'GET' HTTP, avec les données codées comme une séquence `clé=valeur `` séparées par des paires de caractères ``&`.

Ce module permet à pywws de téléverser les données lues à ces organisations. Il est hautement personnalisable en utilisant les fichiers de configuration. Chaque 'service' nécessite un fichier de configuration et de deux gabarits dans `pywws/services` (ne devrait pas avoir besoin d'être modifié par l'utilisateur) et une section dans le fichier `weather.ini` contenant des données spécifiques sur l'utilisateur telles que votre identificateur de site et le mot de passe.

Il existe actuellement six services pour lesquels des fichiers de configuration ont été créés.

| organisation | service name | fichier de configuration |
|---|---|---|
| UK Met Office | `metoffice` | `../../pywws/services/metoffice.ini` |
| Open Weather Map | `openweathermap` | `../../pywws/services/openweathermap.ini` |
| PWS Weather | `pwsweather` | `../../pywws/services/pwsweather.ini` |
| Stacja Pogody | `stacjapogodywawpl` | `../../pywws/services/stacjapogodywawpl.ini` |
| temperatur.nu | `temperaturnu` | `../../pywws/services/temperaturnu.ini` |
| Weather Underground | `underground` | `../../pywws/services/underground.ini` |
| wetter.com | `wetterarchivde` | `../../pywws/services/wetterarchivde.ini` |

### Configuration

Si vous ne l'avez pas déjà fait, visitez le site Web de l'organisation et créez un compte pour votre station météo. Prenez note de tous les détails d'identification du site et du mot de passe vous ayant été donnés.

Arrêtez toute instance du logiciel pywws en fonction puis exécutez `toservice.py` pour créer une section dans le fichier `weather.ini` :

```
python -m pywws.toservice data_dir service_name
```

`service_name` est le nom de services en un seul mot, tels que `metoffice`, `data_dir` est votre répertoire de données météo, comme d'habitude.

Editer le fichier `weather.ini` et trouver la section correspondant au nom du service, par exemple, `[underground]`. Copier les détails de votre site dans cette section, par exemple :

```
[underground]
password = secret
station = ABCDEFG1A
```

Maintenant, vous pouvez tester votre configuration :

```
python -m pywws.toservice -vvv data_dir service_name
```

Celà devrait vous afficher la chaîne de données qui est téléversé. Tout manquement devrait générer un message d'erreur.

### Téléverser les anciennes données

Maintenant, vous pouvez téléverser les données de vos 7 derniers jours, si le service le prend en charge. Modifier votre fichier `status.ini` et supprimer la ligne `last update` de la section appropriée, puis exécutez `toservice.py` avec l'option de rattrapage 'catchup' :

```
python -m pywws.toservice -cvv data_dir service_name
```

Cela peut prendre 20 minutes ou plus, en fonction de la quantité de données que vous avez.

#### Ajoutez le(s) téléversement de(s) service(s) aux tâches régulières

Modifiez votre fichier `weather.ini` de nouveau et ajoutez une liste de services dans la section `[live]`, `[logged]`, `[hourly]`, `[12 hourly]` ou `[daily]`, en fonction de la fréquence à laquelle vous souhaitez envoyer les données. Par exemple :

```
[live]
twitter = []
plot = []
text = []
services = ['underground_rf']

[logged]
twitter = []
plot = []
text = []
services = ['metoffice', 'stacjapogodywawpl']

[hourly]
twitter = []
plot = []
text = []
services = ['underground']
```

Notez que la section `[live]` n'est utilisée que lors de l'exécution de *LiveLog*. C'est une bonne idée de répéter tout service sélectionné dans `[live]` dans la section `[logged]` ou `[hourly]` au cas où vous passez au fonctionnement par *Hourly*.

Redémarrez votre programme pywws régulier (*Hourly* ou *LiveLog*) et visitez le site Web approprié pour voir les mises à jour régulières de votre station météo.

#### Notes sur les services

**UK Met Office**

- Créer un compte : https://register.metoffice.gov.uk/WaveRegistrationClient/public/register.do?service=weatherobservations
- API : http://wow.metoffice.gov.uk/support?category=dataformats#automatic
- Exemple `weather.ini` section :

```
[metoffice]
site id = 12345678
aws pin = 987654
```

**Open Weather Map**

- Créer un compte : http://openweathermap.org/login
- API : http://openweathermap.org/API
- Exemple `weather.ini` section :

```
[openweathermap]
lat = 51.501
long = -0.142
alt = 10
user = Elizabeth Windsor
password = corgi
id = Buck House
```

Le comportement par défaut est d'utiliser votre nom d'utilisateur pour identifier la station météo. Cependant, il est possible pour un utilisateur d'avoir plus d'une station météo, il y a donc un paramètre''name'' dans l'API qui peut être utilisé pour identifier la station. Cela apparaît comme `id` dans `weather.ini`. Assurez-vous que vous ne choisissez pas un nom qui est déjà utilisé.

**PWS Weather**

— Créer un compte : http://www.pwsweather.com/register.php
— API basé sur le protocol WU : http://wiki.wunderground.com/index.php/PWS_-_Upload_Protocol
— Exemple `weather.ini` section :

```
[pwsweather]
station = ABCDEFGH1
password = xxxxxxx
```

**Weather Underground**

— Créer un compte : http://www.wunderground.com/members/signup.asp
— API : http://wiki.wunderground.com/index.php/PWS_-_Upload_Protocol
— Exemple `weather.ini` section :

```
[underground]
station = ABCDEFGH1
password = xxxxxxx
```

### API

#### Fonctions

| | |
|---|---|
| *main*([argv]) | |

#### Classes

| | |
|---|---|
| *ToService*(params, status, calib_data, ...) | Téléverse les données météo pour les services météorologiques tels que Weather Under |

**class** pywws.toservice.**ToService**(*params*, *status*, *calib_data*, *service_name*)

Téléverse les données météo pour les services météorologiques tels que Weather Underground.

**Paramètres**
—**params** (*pywws.DataStore.params*) – pywws configuration.
—**status** (*pywws.DataStore.status*) – pywws status store.
—**calib_data** (*pywws.DataStore.calib_store*) – 'calibrated' data.
—**service_name** (*string*) – name of service to upload to.

**encode_data**(*data*)

Encode un enregistrement de données météo.

Le paramètre `data` contient les données à être encodées. Il devrait y avoir un enregistrement 'calibré' de données, tels que stockés dans *pywws.DataStore.calib_store*.

**Paramètres****data** (*dict*) – the weather data record.

**Retourne**urlencoded data.

**Type retourné**string

**send_data**(*coded_data*)

Téléverse un enregistrement de données météo.

Le paramètre `data` contient les données à être téléversées. Il doit être une chaîne urlencodée .

**Paramètres** **coded_data** – the data to upload.

**Retourne** success status

**Type retourné** bool

**next_data**(*start*, *live_data*)

Obtien les enregistrements de données météorologiques à téléverser.

Cette méthode téléverse le plus récent enregistrement de données météo ou tous les enregistrements depuis une date fournie, selon la valeur de `start`.

**Paramètres** **start** – datetime of first record to get, or None to get

données les plus récentes seulement.

**Paramètres** **live_data** (*dict*) – a current 'live' data record, or None.

**Retourne** yields weather data records.

**Type retourné** dict

**catchup_start**()

Obtien l'horodatage du premier enregistrement de rattrapage (catchup) envoyé

**Type retourné** datetime

**Upload**(*catchup*, *live_data=None*)

Téléverse un ou plusieurs enregistrement(s) de données météo.

Cette méthode téléverse le plus récent enregistrement de données météo ou tous les enregistrements depuis le dernier téléchargement (jusqu'à 7 jours), selon la valeur de `catchup`.

Il définit la valeur de configuration `last update` à l'horodatage de l'enregistrement le plus récent correctement téléversé.

**Paramètres** **catchup** (*bool*) – upload all data since last upload.

**Retourne** success status

**Type retourné** bool

pywws.toservice.**main**(*argv=None*)

### pywws.YoWindow

Génére un fichier XML YoWindow.

```
usage: python -m pywws.YoWindow [options] data_dir output_file
options are:
 -h or --help     display this help
 -v or --verbose  increase amount of reassuring messages
data_dir is the root directory of the weather data
output_file is the YoWindow XML file to be written
```

#### Fonctions

| | |
|---|---|
| *main*([argv]) | |

#### Classes

| | |
|---|---|
| *YoWindow*(calib_data) | Classe pour écrire un fichier XML YoWindow. |

**class** pywws.YoWindow.**YoWindow**(*calib_data*)

Classe pour écrire un fichier XML YoWindow. Pour voir les spécifications du fichier http://yowindow.com/doc/yowindow_pws_format.xml

**write_file**(*file_name*, *data=None*)

pywws.YoWindow.**main**(*argv=None*)

### pywws.WeatherStation

Obtient les données des stations météorologiques WH1080/WH3080 et compatibles.

Dérivé de wwsr.c par Michael Pendec (michael.pendec@gmail.com), wwsrdump.c par Svend Skafte (svend@skafte.net), modifié par Dave Wells, et d'autres sources.

#### Introduction

C'est le module qui parle à l'unité de base de la station météo. Je n'ai pas beaucoup de compréhension de l'USB, donc j'ai copié beaucoup du programme C wwsr de Michael Pendec.

La mémoire de la station météorologique comporte deux parties : un "bloc fixe" de 256 octets et une mémoire tampon circulaire de 65280 octets. Comme chaque lecture météo prend 16 octets, la station peut stocker 4080 lectures, ou 14 jours de lectures intervalle de 5 minutes. (Les stations de type 3080 stockent 20 octets par la lecture, pour un maximum de 3264 lectures) Comme les données sont lues en bloc de 32 octets, mais chaque lecture météorologique est de 16 ou 20 octets, un petit cache est utilisé pour réduire le trafic USB. Le comportement de mise en cache mémoire peut être contourné avec le paramètre `unbuffered` de `get_data` et `get_raw_data`.

Le décodage des données est contrôlé par les dictionnaires statiques `reading_format`, `lo_fix_format`, `fixed_format`. Les clés sont les noms des éléments de données et les valeurs peuvent être un tulpe `(décalage, type, multiplicateur)` `` ou un autre dictionnaire. Ainsi, par exemple, l'entrée du dictionnaire ``reading_format 'rain' : (13, 'us', 0.3)` signifie que la valeur de 'rain' est un entier court non signé (deux octets), à 13 octets à partir du début du bloc, et devrait être multiplié par 0,3 à obtenir une valeur utile.

L'utilisation de dictionnaires imbriqués dans le dictionnaire `fixed_format` permet de décoder des sous-ensembles de données utiles . Par exemple, pour décoder le bloc entier `get_fixed_block` celui-ci est appelée sans paramètres :

```
ws = WeatherStation.weather_station()
print ws.get_fixed_block()
```

Pour obtenir la température extérieure minimale stockées, `get_fixed_block` est appelée avec une séquence de clés :

```
ws = WeatherStation.weather_station()
print ws.get_fixed_block(['min', 'temp_out', 'val'])
```

Souvent, il n'est pas nécessaire de lire et décoder l'ensemble du bloc fixe, comme ses 64 premiers octets contiennent les données les plus utiles : l'intervalle entre les lectures enregistrées, l'adresse du tampon où la lecture en cours est mémorisée, ainsi que la date et l'heure courante. La méthode `get_lo_fix_block` offre un accès facile à ces données.

Pour d'autres exemples de l'utilisation du module WeatherStation, voir le programme TestWeatherStation.

### API détaillé

### Fonctions

| | |
|---|---|
| *decode_status*(status) | |

### Classes

| | |
|---|---|
| *CUSBDrive*() | Interface de bas niveau de la station météo via USB. |
| *weather_station*([ws_type, params, status]) | Classe qui représente la station météo pour le programme utilisateur. |

pywws.WeatherStation.**decode_status**(*status*)

**class** pywws.WeatherStation.**CUSBDrive**

Interface de bas niveau de la station météo via USB.

Vaguement calqué sur une classe C++ obtenue à partir http://site.ambientweatherstore.com/easyweather/ws_1080_2080_protocol.
Je n'en sais pas la provenance, mais il semble que cela pourrait provenir du fabricant.

**EndMark = 32**

**ReadCommand = 161**

**WriteCommand = 160**

**WriteCommandWord = 162**

**read_block**(*address*)

Lit 32 octets à partir de la station météo.

Si la lecture échoue pour une raison quelconque, None est retourné.

**Paramètres** **address** (*int*) – address to read from.

**Retourne** the data from the weather station.

**Type retourné** list(int)

**write_byte**(*address*, *data*)

Écrire un seul octet à la station météo.

**Paramètres**

—**address** (*int*) – address to write to.

—**data** (*int*) – the value to write.

**Retourne** success status.

**Type retourné** bool

**class** pywws.WeatherStation.**weather_station**(*ws_type='1080'*, *params=None*, *status=None*)

Classe qui représente la station météo pour le programme utilisateur.

Connecte à la station météo et prépare à lire les données.

**avoid = 3.0**

**min_pause = 0.5**

**live_data**(*logged_only=False*)

**inc_ptr**(*ptr*)

Retourne le pointeur de données suivant du tampon circulaire .

**dec_ptr**(*ptr*)

Obtient le pointeur de donnée du tampon circulaire précédent.

**get_raw_data**(*ptr*, *unbuffered=False*)

Obtient les données brutes à partir d'un tampon circulaire.

Si 'unbuffered' est faux, une valeur mise en cache, obtenu précédemment, peut être retourné.

**get_data**(*ptr*, *unbuffered=False*)
Obtenir les données décodées à partir du tampon circulaire.

Si 'unbuffered' est faux, une valeur mise en cache, obtenu précédemment, peut être retourné.

**current_pos**()
Obtenir l'emplacement du tampon circulaire lorsque les données actuelles sont en cours d'écriture.

**get_raw_fixed_block**(*unbuffered=False*)
Obtenir le "bloc fixe" brut des paramètres et des données MIN/MAX.

**get_fixed_block**(*keys=[]*, *unbuffered=False*)
Obtient le "bloc fixe" décodé des paramètres et des données MIN/MAX.

Un sous-ensemble du bloc complet pouvant être sélectionné par clés.

**write_data**(*data*)
Écrit un ensemble d'octets unique vers la station météo. Les données doivent être un tableau de (ptr, valeur) paires.

**reading_format = {'3080' : {'status' : (15, 'pb', None), 'hum_out' : (4, 'ub', None), 'wind_gust' : (10, 'wg', 0.1), 'uv' :**

**lo_fix_format = {'alarm_1' : (21, 'bf', ('bit0', 'time', 'wind_dir', 'bit3', 'hum_in_lo', 'hum_in_hi', 'hum_out_lo', 'hu**

**fixed_format = {'alarm_1' : (21, 'bf', ('bit0', 'time', 'wind_dir', 'bit3', 'hum_in_lo', 'hum_in_hi', 'hum_out_lo', 'hum**

**data_start = 256**

**reading_len = {'3080' : 20, '1080' : 16}**

### pywws.device_ctypes_hidapi

Low level USB interface to weather station, using ctypes to access hidapi.

#### Introduction

This module handles low level communication with the weather station via ctypes and the hidapi library. Alternative modules, pywws.device_cython_hidapi and pywws.device_pyusb, use other libraries. The choice of which module to use depends on which libraries are available for you computer.

Les utilisateurs de versions récentes de Mac OS n'ont ont moins de choix. Le système d'exploitation rend très difficile l'accès direct aux périphériques HID (comme la station météo), de sorte que la bibliothèque `hidapi` doit être utilisée.

Users of OpenWRT and similar embedded Linux platforms will probably not be able to install `ctypes` or `cython-hidapi`, so are constrained to use `libusb` and its `PyUSB` Python interface.

#### Installation

Certains de ces logiciels peuvent être déjà installés sur votre machine, il faut vérifier avant de télécharger les sources et les compiler vous-même.

1. Installation de hidapi.
   Créez une copie locale du dépôt git, placez vous dans ce nouveau répertoire, puis suivez les instructions du fichier `README.txt` :

```
git clone https://github.com/signal11/hidapi.git
cd hidapi
more README.txt
```

2. Install ctypes.
   Ceux-ci devraient être disponibles sous forme de paquets pour votre système d'exploitation. Par exemple :

```
sudo zypper install python-ctypes
```

#### Vérification

Exécute `TestWeatherStation.py` avec un niveau de message accru de sorte qu'il indique quel module d'accès au dispositif USB est utilisé :

```
python TestWeatherStation.py -vv
18:10:27:pywws.WeatherStation.CUSBDrive:using pywws.device_ctypes_hidapi
0000 55 aa ff ff ff ff ff ff ff ff ff ff ff ff ff 05 20 01 51 11 00 00 00 81 00 00 07 01 00 d0 56
0020 61 1c 61 1c 00 00 00 00 00 00 00 12 02 14 18 09 41 23 c8 00 32 80 47 2d 2c 01 2c 81 5e 01 1e 80
0040 a0 00 c8 80 a0 28 80 25 a0 28 80 25 03 36 00 05 6b 00 00 0a 00 f4 01 18 00 00 00 00 00 00 00 00
0060 00 00 54 1c 63 0a 2f 01 71 00 7a 01 59 80 7a 01 59 80 e4 00 f5 ff 69 54 00 00 fe ff 00 00 b3 01
0080 0c 02 d0 ff d3 ff 5a 24 d2 24 dc 17 00 11 09 06 15 40 10 03 07 22 18 10 08 11 08 30 11 03 07 12
00a0 36 08 07 24 17 17 11 02 28 10 10 09 06 30 14 29 12 02 11 06 57 09 06 30 14 29 12 02 11 06 57 08
00c0 08 31 14 30 12 02 14 18 04 12 02 01 10 12 11 09 13 17 19 11 08 21 16 53 11 09 13 17 19 12 01 18
00e0 07 17 10 02 22 11 06 11 11 06 13 12 11 11 06 13 12 11 11 10 11 38 11 11 10 11 38 10 02 22 14 43
```

#### API

#### Fonctions

---

#### Classes

| | |
|---|---|
| *USBDevice*(vendor_id, product_id) | Low level USB device access via hidapi library. |

**class** pywws.device_ctypes_hidapi.**USBDevice**(*vendor_id*, *product_id*)

Low level USB device access via hidapi library.

**Paramètres**

—**idVendor** (*int*) – the USB "vendor ID" number, for example 0x1941.

—**idProduct** (*int*) – the USB "product ID" number, for example 0x8021.

**read_data**(*size*)

Reçoit des données de l'appareil.

Si la lecture échoue pour une raison quelconque, une exception `IOError` est levée.

**Paramètres****size** (*int*) – the number of bytes to read.

**Retourne**the data received.

**Type retourné**list(int)

**write_data**(*buf*)

Envoyer donnée au service

**Paramètres****buf** (*list(int)*) – the data to send.

**Retourne**success status.

**Type retourné**bool

### pywws.device_cython_hidapi

Interface USB de bas niveau de la station météo, utilisant cython-hidapi.

### Introduction

Ce module gère la communication de bas niveau avec la station météo via la bibliothèque cython-hidapi . Un module alternatif, pywws.device_pyusb, utilise la bibliothèque PyUSB. Le choix du module à utiliser dépend de la disponibilité des bibliothèques pour votre ordinateur.

Les utilisateurs de versions récentes de Mac OS n'ont pas le choix. Le système d'exploitation rend très difficile l'accès direct aux périphériques HID (comme la station météo), de sorte que la bibliothèque `hidapi` doit être utilisée. `Cython-hidapi` est une interface Python pour cette bibliothèque.

Les utilisateurs de OpenWRT et plateformes embarquées Linux similaires ne seront probablement pas en mesure d'installer `cython-hidapi` et sont donc contraints d'utiliser `libusb` et son interface Python `PyUSB`.

### Installation

Certains de ces logiciels peuvent être déjà installés sur votre machine, il faut vérifier avant de télécharger les sources et les compiler vous-même.

1. Installation de hidapi.
   Créez une copie locale du dépôt git, placez vous dans ce nouveau répertoire, puis suivez les instructions du fichier `README.txt` :

```
git clone https://github.com/signal11/hidapi.git
cd hidapi
more README.txt
```

2. Installation cython.
   Ceux-ci devraient être disponibles sous forme de paquets pour votre système d'exploitation. Par exemple :

```
sudo apt-get install cython
```

3. Install cython-hidapi.
   Ceci doit aussi être téléchargé et compilé :

```
git clone https://github.com/gbishop/cython-hidapi.git
cd cython-hidapi
python setup.py build
sudo python setup.py install
```

   Remplacez `setup.py` par `setup-mac.py` ou `setup-windows.py` si vous utilisez un Mac OS ou Windows.

### Vérification

Exécute `TestWeatherStation.py` avec un niveau de message accru de sorte qu'il indique quel module d'accès au dispositif USB est utilisé :

```
python TestWeatherStation.py -vv
18:10:27:pywws.WeatherStation.CUSBDrive:using pywws.device_cython_hidapi
0000 55 aa ff ff ff ff ff ff ff ff ff ff ff ff ff ff 05 20 01 51 11 00 00 00 81 00 00 07 01 00 d0 56
0020 61 1c 61 1c 00 00 00 00 00 00 00 12 02 14 18 09 41 23 c8 00 32 80 47 2d 2c 01 2c 81 5e 01 1e 80
0040 a0 00 c8 80 a0 28 80 25 a0 28 80 25 03 36 00 05 6b 00 00 0a 00 f4 01 18 00 00 00 00 00 00 00 00
0060 00 00 54 1c 63 0a 2f 01 71 00 7a 01 59 80 7a 01 59 80 e4 00 f5 ff 69 54 00 00 fe ff 00 00 b3 01
0080 0c 02 d0 ff d3 ff 5a 24 d2 24 dc 17 00 11 09 06 15 40 10 03 07 22 18 10 08 11 08 30 11 03 07 12
00a0 36 08 07 24 17 17 11 02 28 10 10 09 06 30 14 29 12 02 11 06 57 09 06 30 14 29 12 02 11 06 57 08
00c0 08 31 14 30 12 02 14 18 04 12 02 01 10 12 11 09 13 17 19 11 08 21 16 53 11 09 13 17 19 12 01 18
00e0 07 17 10 02 22 11 06 11 11 06 13 12 11 11 06 13 12 11 11 10 11 38 11 11 10 11 38 10 02 22 14 43
```

#### API

#### Classes

| | |
|---|---|
| *USBDevice*(idVendor, idProduct) | Accès de bas niveau au périphérique USB via la bibliothèque cython-hidapi. |

**class** pywws.device_cython_hidapi.**USBDevice**(*idVendor*, *idProduct*)

Accès de bas niveau au périphérique USB via la bibliothèque cython-hidapi.

**Paramètres**

—**idVendor** (*int*) – the USB "vendor ID" number, for example 0x1941.

—**idProduct** (*int*) – the USB "product ID" number, for example 0x8021.

**read_data**(*size*)

Reçoit des données de l'appareil.

Si la lecture échoue pour une raison quelconque, une exception IOError est levée.

**Paramètres****size** (*int*) – the number of bytes to read.

**Retourne**the data received.

**Type retourné**list(int)

**write_data**(*buf*)

Envoyer donnée au service

**Paramètres****buf** (*list(int)*) – the data to send.

**Retourne**success status.

**Type retourné**bool

### pywws.device_pyusb1

Interface USB de bas niveau de la station météo, utilisant PyUSB.

#### Introduction

Ce module gère la communication de bas niveau avec la station météo via la bibliothèque PyUSB . Les modules alternatifs, pywws.device_pyusb, pywws.device_ctypes_hidapi, and pywws.device_cython_hidapi, utilisent différentes bibliothèques. Le choix du module à utiliser dépend de la disponibilité des bibliothèques selon votre ordinateur.

Les utilisateurs de versions récentes de Mac OS n'ont ont moins de choix. Le système d'exploitation rend très difficile l'accès direct aux périphériques HID (comme la station météo), de sorte que la bibliothèque hidapi doit être utilisée.

#### Installation

Certains de ces logiciels peuvent être déjà installés sur votre machine, il faut vérifier avant de télécharger les sources et les compiler vous-même.

1. Installer libusb et PyUSB.

   Ceux-ci devraient être disponibles sous forme de paquets pour votre système d'exploitation, mais leurs noms peuvent varier. Par exemple, sous Ubuntu Linux :

```
sudo apt-get install python-usb
```

   Sur certains systèmes embarqués sous Linux :

```
ipkg install libusb py25-usb
```

#### Vérification

Exécute `TestWeatherStation.py` avec un niveau de message accru de sorte qu'il indique quel module d'accès au dispositif USB est utilisé :

```
python TestWeatherStation.py -vv
18:28:09:pywws.WeatherStation.CUSBDrive:using pywws.device_pyusb1
0000 55 aa ff ff ff ff ff ff ff ff ff ff ff ff ff ff 05 20 01 41 11 00 00 00 81 00 00 0f 05 00 e0 51
0020 03 27 ce 27 00 00 00 00 00 00 00 12 02 14 18 27 41 23 c8 00 00 00 46 2d 2c 01 64 80 c8 00 00 00
0040 64 00 64 80 a0 28 80 25 a0 28 80 25 03 36 00 05 6b 00 00 0a 00 f4 01 12 00 00 00 00 00 00 00 00
0060 00 00 49 0a 63 12 05 01 7f 00 36 01 60 80 36 01 60 80 bc 00 7b 80 95 28 12 26 6c 28 25 26 c8 01
0080 1d 02 d8 00 de 00 ff 00 ff 00 ff 00 00 11 10 06 01 29 12 02 01 19 32 11 09 09 05 18 12 01 22 13
00a0 14 11 11 04 15 04 11 12 17 05 12 11 09 02 15 26 12 02 11 07 05 11 09 02 15 26 12 02 11 07 05 11
00c0 09 10 09 12 12 02 02 12 38 12 02 07 19 00 11 12 16 03 27 12 02 03 11 00 11 12 16 03 27 11 12 26
00e0 21 32 11 12 26 21 32 12 02 06 19 57 12 02 06 19 57 12 02 06 19 57 12 02 06 19 57
```

#### API

#### Classes

| | |
|---|---|
| *USBDevice*(idVendor, idProduct) | Accès de bas niveau au périphérique USB via la bibliothèque PyUSB 1.0. |

**class** pywws.device_pyusb1.**USBDevice**(*idVendor*, *idProduct*)

Accès de bas niveau au périphérique USB via la bibliothèque PyUSB 1.0.

**Paramètres**

—**idVendor** (*int*) – the USB "vendor ID" number, for example 0x1941.

—**idProduct** (*int*) – the USB "product ID" number, for example 0x8021.

**read_data**(*size*)

Reçoit des données de l'appareil.

Si la lecture échoue pour une raison quelconque, une exception `IOError` est levée.

**Paramètres**size (*int*) – the number of bytes to read.

**Retourne**the data received.

**Type retourné**list(int)

**write_data**(*buf*)

Envoyer donnée au service

Si l'écriture échoue pour une raison quelconque, une exception `IOError` est levée.

**Paramètres**buf (*list(int)*) – the data to send.

**Retourne**success status.

**Type retourné**bool

### pywws.device_pyusb

Interface USB de bas niveau de la station météo, utilisant PyUSB.

### Introduction

Ce module gère la communication de bas niveau avec la station météo via la bibliothèque PyUSB . Un module alternatif, pywws.device_cython_hidapi, utilise la bibliothèque cython-hidapi. Le choix du module à utiliser dépend de la disponibilité des bibliothèques selon votre ordinateur.

Les utilisateurs de versions récentes de Mac OS n'ont pas le choix. Le système d'exploitation rend très difficile l'accès direct aux périphériques HID (comme la station météo), de sorte que la bibliothèque `hidapi` doit être utilisée. `Cython-hidapi` est une interface Python pour cette bibliothèque.

Les utilisateurs de OpenWRT et plateformes embarquées Linux similaires ne seront probablement pas en mesure d'installer `cython-hidapi` et sont donc contraints d'utiliser `libusb` et son interface Python `PyUSB`.

### Installation

Certains de ces logiciels peuvent être déjà installés sur votre machine, il faut vérifier avant de télécharger les sources et les compiler vous-même.

1. Installer libusb et PyUSB.

   Ceux-ci devraient être disponibles sous forme de paquets pour votre système d'exploitation, mais leurs noms peuvent varier. Par exemple, sous Ubuntu Linux :

```
sudo apt-get install libusb-0.1 python-usb
```

   Sur certains systèmes embarqués sous Linux :

```
ipkg install libusb py25-usb
```

### Vérification

Exécute `TestWeatherStation.py` avec un niveau de message accru de sorte qu'il indique quel module d'accès au dispositif USB est utilisé :

```
python TestWeatherStation.py -vv
18:28:09:pywws.WeatherStation.CUSBDrive:using pywws.device_pyusb
0000 55 aa ff ff ff ff ff ff ff ff ff ff ff ff ff ff 05 20 01 41 11 00 00 00 81 00 00 0f 05 00 e0 51
0020 03 27 ce 27 00 00 00 00 00 00 00 12 02 14 18 27 41 23 c8 00 00 00 46 2d 2c 01 64 80 c8 00 00 00
0040 64 00 64 80 a0 28 80 25 a0 28 80 25 03 36 00 05 6b 00 00 0a 00 f4 01 12 00 00 00 00 00 00 00 00
0060 00 00 49 0a 63 12 05 01 7f 00 36 01 60 80 36 01 60 80 bc 00 7b 80 95 28 12 26 6c 28 25 26 c8 01
0080 1d 02 d8 00 de 00 ff 00 ff 00 ff 00 00 11 10 06 01 29 12 02 01 19 32 11 09 09 05 18 12 01 22 13
00a0 14 11 11 04 15 04 11 12 17 05 12 11 09 02 15 26 12 02 11 07 05 11 09 02 15 26 12 02 11 07 05 11
00c0 09 10 09 12 12 02 02 12 38 12 02 07 19 00 11 12 16 03 27 12 02 03 11 00 11 12 16 03 27 11 12 26
00e0 21 32 11 12 26 21 32 12 02 06 19 57 12 02 06 19 57 12 02 06 19 57 12 02 06 19 57 12 02 06 19 57
```

### API

### Classes

| | |
|---|---|
| *USBDevice*(idVendor, idProduct) | Accès de bas niveau au périphérique USB via la bibliothèque PyUSB. |

**class** pywws.device_pyusb.**USBDevice**(*idVendor*, *idProduct*)

Accès de bas niveau au périphérique USB via la bibliothèque PyUSB.

**Paramètres**

—**idVendor** (*int*) – the USB "vendor ID" number, for example 0x1941.
—**idProduct** (*int*) – the USB "product ID" number, for example 0x8021.

**read_data**(*size*)
Reçoit des données de l'appareil.
Si la lecture échoue pour une raison quelconque, une exception `IOError` est levée.

**Paramètres** **size** (*int*) – the number of bytes to read.

**Retourne** the data received.

**Type retourné** list(int)

**write_data**(*buf*)
Envoyer donnée au service
Si l'écriture échoue pour une raison quelconque, une exception `IOError` est levée.

**Paramètres** **buf** (*list(int)*) – the data to send.

**Retourne** success status.

**Type retourné** bool

### pywws.DataStore

DataStore.py - enregistre les lectures dans des fichiers facilement accessibles

#### Introduction

Ce module est au cœur de mon logiciel de station météo. Il stocke les données sur disque, mais sans le coût d'un système de base de données à grande échelle. Je l'ai conçu pour fonctionner sur une machine avec peu de mémoire comme mon routeur Asus. Pour minimiser l'utilisation de la mémoire, il ne charge que l'équivalent d'une journée de données à la fois dans la mémoire.

D'un point de vue "utilisateur", les données sont accédées comme un croisement entre une liste et un dictionnaire. Chaque enregistrement de données est indexée par un objet `datetime.datetime` (comportement dictionnaire), mais les enregistrements sont stockés dans l'ordre et peut être consulté sous forme de tranches (comportement liste).

Par exemple, pour accéder aux données horaires pour le jour de Noël 2009, on peut faire ce qui suit

```
from datetime import datetime
import DataStore
hourly = DataStore.hourly_store('weather_data')
for data in hourly[datetime(2009, 12, 25):datetime(2009, 12, 26)]:
    print data['idx'], data['temp_out']
```

D'autres exemples d'accès aux données :

```
# get value nearest 9:30 on Christmas day 2008
data[data.nearest(datetime(2008, 12, 25, 9, 30))]
# get entire array, equivalent to data[:]
data[datetime.min:datetime.max]
# get last 12 hours worth of data
data[datetime.utcnow() - timedelta(hours=12):]
```

Le module fournit cinq classes pour stocker des données différentes. *data_store* prend les données "brutes" de la station météo ; *calib_store*, *hourly_store*, *daily_store* et *monthly_store* stockent les données traitées (voir *pywws.Process*). Les trois sont dérivés de la même classe `core_store`, ils ne diffèrent que par les clés et les types de données stockées dans chaque enregistrement.

### API détaillé

### Fonctions

| | |
|---|---|
| safestrptime(date_string[, format]) | |

### Classes

| | |
|---|---|
| ParamStore(root_dir, file_name) | |
| calib_store(root_dir) | Stocke les données “calibrées” de la station météo. |
| core_store(root_dir) | |
| daily_store(root_dir) | Stocke les données sommaires quotidiens de la stations météo. |
| data_store(root_dir) | Stocke les données brutes de la station météo. |
| hourly_store(root_dir) | Stocke les données sommaires horaire de la stations météo. |
| monthly_store(root_dir) | Stocke les données mensuelles sommaire de la stations météo. |
| params(root_dir) | Les paramètres sont stockés dans le fichier “weather.ini”, dans le répertoire spécifié par root_di |
| status(root_dir) | Le status est stocké dans le fichier “status.ini”, dans le répertoire spécifié par root_dir. |

pywws.DataStore.**safestrptime**(*date_string*, *format=None*)

**class** pywws.DataStore.**ParamStore**(*root_dir*, *file_name*)

**flush**()

**get**(*section*, *option*, *default=None*)
Obtient une valeur de paramètre et renvoie une chaîne.

Si la valeur par défaut est spécifiée et la section ou l’option n’est pas définie dans le fichier, ils sont créés et définit par défaut, qui est alors la valeur retournée.

**get_datetime**(*section*, *option*, *default=None*)

**set**(*section*, *option*, *value*)
Définit l’option dans la section, à chaîne.

**unset**(*section*, *option*)
Supprimer l’option de la section.

**class** pywws.DataStore.**params**(*root_dir*)
Les paramètres sont stockés dans le fichier “weather.ini”, dans le répertoire spécifié par root_dir.

**class** pywws.DataStore.**status**(*root_dir*)
Le status est stocké dans le fichier “status.ini”, dans le répertoire spécifié par root_dir.

**class** pywws.DataStore.**core_store**(*root_dir*)

**before**(*idx*)
Retourne datetime (horodate) du plus récent enregistrement de données existant dont datetime est <idx.

Peut même ne pas être dans la même année ! Si aucun enregistrement n’existe, retourne None (Aucun).

**after**(*idx*)
Retourne datetime (horodate) de la plus ancienne donnée existante dont datetime est> = idx.

Peut même ne pas être dans la même année ! Si aucun enregistrement n’existe, retourne None (Aucun).

**nearest**(*idx*)
Retourne le datetime (horodate) de l’enregistrement dont le datetime est le plus près de idx.

**flush**()

**class** pywws.DataStore.**data_store**(*root_dir*)
Stocke les données brutes de la station météo.
**key_list = ['idx', 'delay', 'hum_in', 'temp_in', 'hum_out', 'temp_out', 'abs_pressure', 'wind_ave', 'wind_gust', 'wind_**
**conv = {'status' : <type 'int'>, 'wind_ave' : <type 'float'>, 'rain' : <type 'float'>, 'hum_in' : <type 'int'>, 'temp_out' : <ty**

**class** pywws.DataStore.**calib_store**(*root_dir*)
Stocke les données "calibrées" de la station météo.
**key_list = ['idx', 'delay', 'hum_in', 'temp_in', 'hum_out', 'temp_out', 'abs_pressure', 'rel_pressure', 'wind_ave', 'win**
**conv = {'status' : <type 'int'>, 'wind_ave' : <type 'float'>, 'rain' : <type 'float'>, 'rel_pressure' : <type 'float'>, 'hum_in'**

**class** pywws.DataStore.**hourly_store**(*root_dir*)
Stocke les données sommaires horaire de la stations météo.
**key_list = ['idx', 'hum_in', 'temp_in', 'hum_out', 'temp_out', 'abs_pressure', 'rel_pressure', 'pressure_trend', 'wind_**
**conv = {'pressure_trend' : <type 'float'>, 'wind_ave' : <type 'float'>, 'rain' : <type 'float'>, 'rel_pressure' : <type 'float'>**

**class** pywws.DataStore.**daily_store**(*root_dir*)
Stocke les données sommaires quotidiens de la stations météo.
**key_list = ['idx', 'start', 'hum_out_ave', 'hum_out_min', 'hum_out_min_t', 'hum_out_max', 'hum_out_max_t', 'temp**
**conv = {'temp_in_min' : <type 'float'>, 'temp_in_max' : <type 'float'>, 'uv_ave' : <type 'float'>, 'temp_out_max' : <type**

**class** pywws.DataStore.**monthly_store**(*root_dir*)
Stocke les données mensuelles sommaire de la stations météo.
**key_list = ['idx', 'start', 'hum_out_ave', 'hum_out_min', 'hum_out_min_t', 'hum_out_max', 'hum_out_max_t', 'temp**
**conv = {'uv_ave' : <type 'float'>, 'illuminance_max_hi_t' : <function safestrptime at 0x7f671bf02cf8>, 'uv_max_lo_t' : <**

### pywws.TimeZone

Fournit un couple d'objets `datetime.tzinfo` représentant l'heure locale et UTC.

#### Introduction

Ce module fournit deux objets `datetime.tzinfo` représentant l'heure UTC et fuseaux horaires locaux. Ceux-ci sont utilisés pour convertir vers et depuis l'heure UTC et l'heure locale. Les logiciels de stations météorologiques emmagasinent les données horodatées en UTC, pour éviter les problèmes avec l'heure d'été, mais le gabarit et les données des graphes sont affichés avec les heures locales.

Le module est copié directement de la documentation du module `datetime.tzinfo`.

#### API détaillé

#### Classes

| | |
|---|---|
| *LocalTimezone* | Local time |
| *UTC* | |

**class** pywws.TimeZone.**UTC**

**utcoffset**(*dt*)
**tzname**(*dt*)
**dst**(*dt*)

**class** pywws.TimeZone.**LocalTimezone**

Local time

**utcoffset**(*dt*)

**dst**(*dt*)

**tzname**(*dt*)

## pywws.Localisation

Localisation.py - fournit des traductions de phrases en langue locale

```
usage: python -m pywws.Localisation [options]
options are:
 -h       or  --help       display this help
 -t code  or  --test code  test use of a language code
```

### Introduction

Certains modules pywws, comme WindRose.py, peuvent automatiquement utiliser la langue locale pour des choses telles que les directions du vent. Le module Localisation.py, surtout copié à partir d'exemples dans la documentation Python, permet cela.

Traduction de pywws se fait en deux parties - traduisant les phrases comme 'rising very rapidly', en changeant le "terme local" qui contrôle des choses comme le noms des mois et la représentation des nombres (par exemple le nombre '23, 2' au lieu de '23 .2 '). Sur certains ordinateurs, il peut ne pas être possible de définir les paramètres régionaux, mais les chaînes traduites peuvent encore être utilisées.

### En utilisant une langue différente

Le langage utilisé par pywws est situé dans la section `[config]` du fichier `weather.ini`. Cela peut être un code de langue à deux lettres, comme `fr` (français), ou pouvez spécifier une variante nationale, telle que `fr_CA` (canadien-français). Il pourrait également s'agir d'un jeu de caractères, par exemple `de_DE.UTF-8`.

Le choix de la langue dépend beaucoup du système, donc Localisation.py peut être exécuté comme un programme autonome pour tester les codes linguistiques. Un bon point de départ pourrait être, par exemple, votre variable d'environnement `LANG` :

```
jim@brains:~/Documents/weather/pywws/code$ echo $LANG
en_GB.UTF-8
jim@brains:~/Documents/weather/pywws/code$ python -m pywws.Localisation -t en_GB.UTF-8
Locale changed from (None, None) to ('en_GB', 'UTF8')
Translation set OK
Locale
  decimal point: 23.2
  date & time: Friday, 14 October (14/10/11 13:02:00)
Translations
  'NNW' => 'NNW'
  'rising very rapidly' => 'rising very rapidly'
  'Rain at times, very unsettled' => 'Rain at times, very unsettled'
jim@brains:~/Documents/weather/pywws/code$
```

Dans la plupart des cas, pas plus que d'un code à deux lettres n'est requis :

```
jim@brains:~/Documents/weather/pywws/code$ python -m pywws.Localisation -t fr
Locale changed from (None, None) to ('fr_FR', 'UTF8')
Translation set OK
Locale
  decimal point: 23,2
  date & time: vendredi, 14 octobre (14/10/2011 13:04:44)
Translations
  'NNW' => 'NNO'
  'rising very rapidly' => 'en hausse très rapide'
  'Rain at times, very unsettled' => 'Quelques précipitations, très perturbé'
jim@brains:~/Documents/weather/pywws/code$
```

Si vous essayez une langue non prise en charge, pywws reprend l'anglais :

```
jim@brains:~/Documents/weather/pywws/code$ python -m pywws.Localisation -t ja
Failed to set locale: ja
No translation file found for: ja
Locale
  decimal point: 23.2
  date & time: Friday, 14 October (10/14/11 13:08:49)
Translations
  'NNW' => 'NNW'
  'rising very rapidly' => 'rising very rapidly'
  'Rain at times, very unsettled' => 'Rain at times, very unsettled'
jim@brains:~/Documents/weather/pywws/code$
```

Une fois que vous avez trouvé un code de langue approprié qui fonctionne, vous pouvez demander à pywws de l'utiliser en éditant votre fichier `weather.ini` :

```
[config]
language = fr
```

#### Créer une nouvelle traduction

S'il n'y a aucun fichier de traduction pour votre langue préférée, alors vous devez en créer un. Voir Comment utiliser pywws dans une autre langue pour obtenir les instructions détaillées.

#### Fonctions

| | |
|---|---|
| *SetApplicationLanguage*(params) | Défini la langue locale et la traduction pour un programme pywws. |
| *SetLocale*(lang) | Régle la langue 'locale' utilisée par un programme. |
| *SetTranslation*(lang) | Régle la traduction utilisée par les modules (certains) pywws. |
| *main*([argv]) | |

pywws.Localisation.**SetLocale**(*lang*)

Régle la langue 'locale' utilisée par un programme.

Cela affecte toute l'application, en changeant la façon dont les dates, les devises et les chiffres sont représentés. Il ne devrait pas être appelée à partir d'une routine de bibliothèque qui peut être utilisée dans un autre programme.

Le paramètre `lang` peut être n'importe quelle chaîne reconnue par `locale.setlocale ()`, par exemple `en`, `fr_FR` ou `fr_FR.UTF-8`.

**Paramètres** **lang** (*string*) – language code.

**Retourne** success status.

**Type retourné**bool

pywws.Localisation.**SetTranslation**(*lang*)

Régle la traduction utilisée par les modules (certains) pywws.

Ceci définit l'objet de la traduction `Localisation.translation` à utiliser une langue particulière.

Le paramètre `lang` peut être une chaîne sous la forme `en`, '' fr_FR'' ou `fr_FR.UTF-8`. Tout ce qui suit un caractère `.` est ignoré. Dans le cas d'une chaîne telle que `fr_FR`, la routine va rechercher un fichier de langue `fr_FR` avant de chercher un `fr`.

**Paramètres** **lang** (*string*) – language code.

**Retourne**success status.

**Type retourné**bool

pywws.Localisation.**SetApplicationLanguage**(*params*)

Défini la langue locale et la traduction pour un programme pywws.

Cette fonction permet de lire la langue à partir du fichier de configuration, puis appel les fonctions *SetLocale()* et *SetTranslation()*.

**Paramètres** **params** (*object*) – a *pywws.DataStore.params* object.

pywws.Localisation.**main**(*argv=None*)

## pywws.calib

### Calibre les données brutes de la station météo

Ce module permet d'ajuster les données brutes de la station météo dans le cadre de l'étape de 'traitement' (voir pywws.Process ). Par exemple, si vous avez installé un entonnoir pour doubler votre zone de collecte du pluviomètre, vous pouvez écrire une routine de calibration pour doubler la valeur de pluie.

**La classe de calibration par défaut fait deux choses :**

1. Générer pression atmosphérique relative.
2. Retirer les valeurs invalides de direction du vent.

Toute calibration utilisateur que vous écrivez doit également faire ceci.

**Écrire votre module de calibration** Tout d'abord, décider où vous voulez garder votre module. Comme vos gabarits texte et graphe, il est préférable de le garder séparé du code pywws, de sorte qu'il n'est pas affecté par les mises à jour de pywws. Je suggère la création d'un répertoire `modules` au même endroit que votre répertoire `templates`.

Créez un fichier texte dans votre répertoire `modules`, par exemple, `Calib.py` et copiez-y le texte suivant :

```
class Calib(object):
    def __init__(self, status):
        self.pressure_offset = eval(status.get('fixed', 'pressure offset'))
    def calib(self, raw):
        result = dict(raw)
        # sanitise data
        if result['wind_dir'] is not None and result['wind_dir'] >= 16:
            result['wind_dir'] = None
        # calculate relative pressure
        result['rel_pressure'] = raw['abs_pressure'] + self.pressure_offset
        return result
```

La classe *Calib* a deux méthodes. `Calib.__init__()` est le constructeur et est un bon endroit pour mettre toutes les constantes nécessaires. :py : meth :*Calib.calib* génère un ensemble unique de données 'calibrées' à partir d'un seul ensemble de données 'brutes'. Il y a quelques règles à suivre lors de l'écriture de cette méthode :

— Assurez-vous d'inclure la ligne `result = dict(raw)`, qui permet de copier toutes les données brutes à votre résultat, au début.
— Ne modifiez pas les données brutes.
— Assurez-vous que vous définissez `result['rel_pressure']`.
— N'oubliez pas de retourner (`return`) le résultat à la fin.

Lorsque vous avez fini d'écrire votre module de calibration vous pouvez demander à pywws de l'utiliser en mettant son emplacement dans votre fichier `weather.ini`. Il va dans les sections `[paths]`, comme le montre l'exemple ci-dessous :

```
[paths]
work = /tmp/weather
templates = /home/jim/weather/templates/
graph_templates = /home/jim/weather/graph_templates/
user_calib = /home/jim/weather/modules/usercalib
```

Notez que la valeur de `user_calib` ne doit pas inclure le `.py` à la fin du nom de fichier.

#### Classes

| | |
|---|---|
| *Calib*(params, status) | Classe qui implémente la calibration par défaut ou la calibration par l'utilisateur. |
| *DefaultCalib*(status) | Classe de calibration par défaut |

**class** pywws.calib.**DefaultCalib**(*status*)

Classe de calibration par défaut

Cette classe définit la pression relative, en utilisant un décalage de pression lues à partir de la station météorologique, et 'normalise' la valeur de la direction du vent. C'est la calibration strictement minimale nécessaire.

**calib**(*raw*)

**class** pywws.calib.**Calib**(*params*, *status*)

Classe qui implémente la calibration par défaut ou la calibration par l'utilisateur.

D'autres modules pywws utilisent cette méthode pour créer un objet de calibration. Le constructeur crée soit un objet de calibration par défaut ou un objet de calibration utilisateur, en fonction de la valeur `user_calib` dans la section `[paths]` du paramètre `params`. Il adopte alors la méthode de calibration de l'objet :py : meth :*calib* comme sien.

**calibrator = None**

### pywws.conversions

conversions.py est un ensemble de fonctions pour convertir les unités pywws natives (centigrades, mm, m / s, hPa), en d'autres unités de mesure populaires

#### Fonctions

| | |
|---|---|
| *apparent_temp*(temp, hr, vent) | Calcule la température apparente (sensation réelle), en utilisant la formule de |
| *cadhumidex*(temp, humidité) | Calcule l'Indice d'Humidité selon les Normes Météo Canadiennes |
| *dew_point*(temp, hum) | Calcule le point de rosée, en utilisant la formule provenant de http://en.wikipedia |
| *illuminance_wm2*(lux) | Conversion approximative de la luminosité en lux, au rayonnement solaire en W/ |
| *pressure_inhg*(hPa) | Converti la pression de hectopascals/millibars, à pouces de mercure |

Suite s

Tableau 4.48 – Suite de la page précédente

| | |
|---|---|
| *pressure_trend_text*(trend) | Converti la tendance de la pression en une chaîne, telle qu'utilisée par le UK Met |
| *rain_inch*(mm) | Converti les précipitations de millimètres, à pouces |
| *temp_f*(c) | Converti la température de Celsius, à Fahrenheit |
| *usaheatindex*(temp, humidité, point de rosée) | Calcule l'Indice de Chaleur selon les normes du Service Météorologique Nationa |
| *wind_bft*(ms) | Converti la vitesse du vent de mètres par seconde, vers l'échelle Beaufort |
| *wind_chill*(temp, vent) | Calcule le refroidissement éolien, en utilisant la formule de |
| *wind_kmph*(ms) | Converti la vitesse du vent de mètres par seconde, à kilomètres par heure |
| *wind_kn*(ms) | Converti la vitesse du vent de mètres par seconde, à noeuds |
| *wind_mph*(ms) | Converti la vitesse du vent de mètres par seconde, en miles par heure |
| *winddir_degrees*(pts) | Converti la direction du vent de 0..15 en degrés |
| *winddir_text*(pts) | Converti la direction du vent de 0..15, en points cardinaux |

pywws.conversions.**illuminance_wm2**(*lux*)
Conversion approximative de la luminosité en lux, au rayonnement solaire en W/m2

pywws.conversions.**pressure_inhg**(*hPa*)
Converti la pression de hectopascals/millibars, à pouces de mercure

pywws.conversions.**pressure_trend_text**(*trend*)
Converti la tendance de la pression en une chaîne, telle qu'utilisée par le UK Met Office.

pywws.conversions.**rain_inch**(*mm*)
Converti les précipitations de millimètres, à pouces

pywws.conversions.**temp_f**(*c*)
Converti la température de Celsius, à Fahrenheit

pywws.conversions.**winddir_degrees**(*pts*)
Converti la direction du vent de 0..15 en degrés

pywws.conversions.**winddir_text**(*pts*)
Converti la direction du vent de 0..15, en points cardinaux

pywws.conversions.**wind_kmph**(*ms*)
Converti la vitesse du vent de mètres par seconde, à kilomètres par heure

pywws.conversions.**wind_mph**(*ms*)
Converti la vitesse du vent de mètres par seconde, en miles par heure

pywws.conversions.**wind_kn**(*ms*)
Converti la vitesse du vent de mètres par seconde, à noeuds

pywws.conversions.**wind_bft**(*ms*)
Converti la vitesse du vent de mètres par seconde, vers l'échelle Beaufort

pywws.conversions.**dew_point**(*temp*, *hum*)
Calcule le point de rosée, en utilisant la formule provenant de http://en.wikipedia.org/wiki/Dew_point.

pywws.conversions.**cadhumidex**(*temp*, *humidity*)
Calcule l'Indice d'Humidité selon les Normes Météo Canadiennes

pywws.conversions.**usaheatindex**(*temp*, *humidity*, *dew*)
Calcule l'Indice de Chaleur selon les normes du Service Météorologique National USA

Voir http://en.wikipedia.org/wiki/Heat_index, formule 1. La formule n'est pas valide pour T < 26.7C, Point de rosée < 12C, ou HR < 40%

pywws.conversions.**wind_chill**(*temp*, *wind*)
Calcule le refroidissement éolien, en utilisant la formule provenant de http://en.wikipedia.org/wiki/wind_chill

pywws.conversions.**apparent_temp**(*temp*, *rh*, *wind*)
Calcule la température apparente (sensation réelle), en utilisant la formule provenant de http://www.bom.gov.au/info/thermal_stress/

### pywws.Logger

Code commun pour l'enregistrement d'informations et d'erreurs.

#### Fonctions

| |
|---|
| *ApplicationLogger*(verbose[, logfile]) |

pywws.Logger.**ApplicationLogger**(*verbose*, *logfile=None*)

## 4.2 Index et tables

— genindex
— modindex
— search

CHAPITRE 5

# Crédits

Je n'aurais pas été en mesure d'obtenir d'informations de la station météorologique sans accès aux sources du programme "wwsr" de Michael Pendec . Je suis également redevable à Dave Wells pour le décodage du "bloc fixe" de données de la station météorologique.

En dernier lieu, un grand vous remercie à tous les utilisateurs de pywws qui ont aidé avec leur questions et suggestions, et particulièrement à ceux qui ont traduit pywws et sa documentation en d'autres langues.

CHAPITRE 6

# Termes

pywws - Logiciel Python pour stations météo USB sans-fil.
http://github.com/jim-easterbrook/pywws

## p

## H



## I



## K



## L



## M



## N



## P

## R



## S



## T



## U



## W

## Y



## Z